Merchant/400™

User's Guide

Version 1.2

COPYRIGHT © 1997 I/NET, INC.

Revision A

Copyright © 1997 I/NET, Inc. All Rights Reserved.

No part of this book may be reproduced in any form or by any means, electronic or mechanical, including photocopying without written permission from I/NET, Inc.

Merchant/400 is a trademark of I/NET, Inc. Other brand and product names are trademarks or registered trademarks of their respective holders.

Table of Contents

1. GETTING STARTED 5

1.1 INSTALLATION 6
1.1.1 Installing Merchant/400 6
1.1.2 Objects Installed 6
1.2 SETTING THE FEATURE ACTIVATION KEY 8
1.3 VIEWING MERCHANT/400 DOCUMENTATION FROM A BROWSER 9
1.3.1 Accessing Document Files Directly 9
1.3.2 Accessing Documentation through Web Server/400 and Commerce Server/400 9

2. GETTING SETUP AS AN INTERNET MERCHANT 10

2.1 GETTING SET UP AS AN INTERNET MERCHANT 11
2.1.1 So, You Want to do Business on the Internet 11
2.1.2 Partner Host Companies That Provide the Link 11
2.2 THE PROCESSING CENTER AND YOUR BANK 12
2.2.1 The Beginning of Becoming an Internet Merchant 12
2.2.2 What You Will Need to Register With a Card Processor 12
2.3 CREDIT CARD NETWORK 14
2.3.1 A Credit Card Transaction House for the World Wide Web 14
2.3.2 Credit Card Network Advertised Features and Benefits 14
2.4 HOLONET IAT 15
2.4.1 Transparent Processing for Credit Card Transactions 15
2.5 ANACOM COMMUNICATIONS, INC. 16
2.5.1 Payment Solutions for Internet Merchants Worldwide 16
2.5.2 Anacom Communications Advertised Features and Benefits 16
2.6 DATACASH LTD. 17
2.6.1 Multi-Currency Payment Solutions for Global Internet Merchants 17
2.6.2 DataCash Advertised Features and Benefits 17
2.7 THE VIRTUAL CREDIT CARD TERMINAL 18
2.7.1 The Connection to the Card Processor 18
2.8 YOUR MERCHANT ID AND PASSWORD 19
2.8.1 Your Keys to Internet Merchant Services 19
2.9 HOW IT ALL FITS TOGETHER 20
2.9.1 The Steps to Take to Become an Internet Merchant 20
2.10 USING MERCHANT/400 WITH NET.COMMERCE 21
2.10.1 Introduction 21
2.10.2 Functionality Provided by the Merchant/400 do_payment Interface 21
2.10.3 Configuring the Merchant/400 Objects for Use with Net.Commerce 23
2.10.4 Creating Your Own API - NCSAMPLE.C and NCSAMPLE2.C 24
2.11 CONFIGURING MERCHANT/400 OBJECTS FOR USE WITH NET.COMMERCE VERSION 2 26
2.11.1 Activating the Customized do_payment API in Net.Commerce version 2 26
2.11.2 Activating the Customized Net.Data macros in Net.Commerce 26
2.12 CONFIGURING MERCHANT/400 OBJECTS FOR USE WITH NET.COMMERCE VERSION 3.2 28
2.12.1 Activating the Customized do_payment API in Net.Commerce version 3.2 28
2.12.2 Activating the Customized Net.Data macros in Net.Commerce 29

3. USING MERCHANT/400 31

3.1 USING THE MERCHANT/400 API 32
3.1.1 Introduction 32
3.1.2 Files Provided to Aid your Development Effort 32
3.2 WHAT IS A MERCHANT SERVER? 34
3.2.1 A Way to do Business Over the Internet 34

3.2.2 The Process ... *34*
3.3 USING THE MERCHANT/400 API IN AN ILE PROGRAM ... 35
3.3.1 Introduction ... *35*
3.3.2 The ILE Call to Execute the API ... *35*
3.4 USING THE MERCHANT/400 API IN AN OPM PROGRAM ... 38
3.4.1 Introduction ... *38*
3.4.2 The OPM Program to Call to Execute the API ... *38*
3.5 INPUT AND OUTPUT FORMATS ... 40
3.5.1 How to Exchange Data With the API. ... *40*
3.6 ERROR CONDITIONS ... 50
3.6.1 What to Expect in the Completion Indicator (plResultSuccess) Parameter ... *50*
3.6.2 plResultSuccess Values. ... *50*
3.7 REQUIRED INFORMATION ... 53
3.7.1 Minimum Data to Send to the API ... *53*
3.8 WHAT HAPPENS - TRANSMISSION METHODS ... 55
3.8.1 The lTransmitMethod / Transmission Method Parameter is the Key ... *55*

4. THE I/NET STORE - A SAMPLE APPLICATION ... 59

4.1 THE I/NET STORE - A SAMPLE APPLICATION ... 60
4.1.1 Introduction ... *60*
4.1.2 Store Screens ... *60*
4.2 CONFIGURATION ... 62
4.2.1 Introduction ... *62*
4.2.2 Running the Sample Store (for Testing Only) ... *62*
4.2.3 Configuring the Sample Store for a Production Environment ... *63*
4.3 THE STORE DATABASE ... 65
4.3.1 Global Configuration Physical File ... *65*
4.3.2 Product Description Physical File ... *66*
4.3.3 Customer Order Physical File ... *67*
4.3.4 Customer Order Item Physical File ... *69*
4.3.5 The Order Identifier Control Physical File ... *69*
4.3.6 Net.Commerce Order Information Physical File ... *69*
4.4 EDITING THE STORE HEADERS AND FOOTERS ... 71
4.4.1 Customizing How the Store Looks ... *71*
4.5 THE STORE MESSAGE FILE ... 73
4.5.1 Store Label Text and Messages ... *73*
4.5.2 Message Groups ... *73*
4.6 HOW THE API FITS IN ... 77
4.6.1 Using the API ... *77*
4.7 COOKIES ... 79
4.7.1 What Is a Cookie? ... *79*
4.7.2 Examples of Using Cookies ... *79*
4.7.3 Advantages and Disadvantages of Using Cookies ... *80*

5. INDEX ... 81

1. Getting Started

1.1 Installation

1.1.1 Installing Merchant/400

1.1.1.1 Important Pre-Installation Instructions

Version 1.0 or greater of Commerce Server/400 (http://www.inetmi.com/products/webserv/webinfo.sht) must be installed on the system before installing Merchant/400 Version 1.0, 1.1 or 1.2. In addition, the following PTFs must also be installed:

- Commerce Server/400 version 1.0C, or PTF number COM100002 (or greater) for Commerce Server/400 version 1.0, 1.0A, or 1.0B
- PTF number WWW130012 for Web Server/400 version 1.3

If Merchant/400 version 1.1 was used in conjunction with Net.Commerce, special care should be taken to migrate existing data to the new format for the MCNINFP file (Net.Commerce Order Information Physical File on page 69). Since the file has changed (to include the DataCash order reference number and credit card start date / issue code fields) it is important to re-duplicate the file using the instructions on the Using Net.Commerce (Configuring the Merchant/400 Objects for Use with Net.Commerce on page 23) page. **Do not forget to retain your historical data by renaming the original files (using RNMOBJ) and copying the data (using CPYF FMTOPT(*MAP *DROP)) to the newly installed and duplicated files.**

1.1.1.2 Installing Merchant/400 requires the following steps:

1. Sign on to your AS/400 as QSECOFR or another user profile with a user class of *SECOFR. Special authorities are required to successfully install Merchant/400 objects (Objects Installed on page 6).
2. End all currently running Web servers with the `ENDWWW` command. Ensure that no product commands or menus are running. Also ensure that the Merchant/400 API is not currently being called by another job (this applies to product re-install only). Use the `WRKACTJOB` command to verify that there are no servers, commands, or other jobs running that may be executing the API.
3. Run the system command **`LODRUN`** to install Merchant/400 from the supplied product tape.

   Note: Prompt the command to change the tape device if the default is not appropriate.

   The install should take approximately 2 to 15 minutes, depending on the tape device and AS/400 model (NOTE: RISC AS/400s may take longer if not using a RISC installation tape due to the extra time it takes to convert IMPI programs to RISC programs). The install program displays a message when it is finished that the install was successful.

   If any problems occur during the install:
   1. Check the joblog to determine the cause of the error.
   2. Correct the problem, if possible.
   3. Re-run the installation command.
   4. If problems persist, please contact support.

1.1.2 Objects Installed

The following objects were created/restored on the AS/400 by the installation program:

- Merchant/400 objects were restored to the Web server's product library (`WWWSERVER`).

- A directory named `WWWServ` was restored into the root of the IFS file system when Commerce Server/400 was installed. `WWWServ` contains many other directories and stream files that are used by Web Server/400 or Commerce Server/400. Merchant/400 publications were restored into the `WWWServ/WebDocs/Shipped/Pubs/Merch400` directory.
- Documents for the sample `I/NET store` application were restored into the `/WWWServ/WebDocs/Shipped/Merchant` directory.
- The library `WWWMRCH` was restored. `WWWMRCH` contains the sample `I/NET Store` application that is used to demonstrate how the API functions. Don't forget to add this library to the list of Script Libraries (with the `ADDWWWSCPL` command) before attempting to run the sample application (section 4.1 on page 60).

  Important note: Merchant/400's sample content and publications will always be shipped in the `Shipped` directory. Users should not place content in this directory for it may be deleted or over-written in the future. In addition, the sample `I/NET Store` application will always be installed into the `WWWMRCH` library and the `/WWWServ/WebDocs/Shipped/Merchant` directory. The contents of these areas should be duplicated by the user to another library and directory for actual use, as they may be deleted or over-written in the future.

1.2 Setting the Feature Activation Key

Merchant/400 uses the activation code facilities built into Commerce Server/400 to set and evaluate the special code provided for the product. If you are reading this documentation online (directly through a network drive), you may want to start the server in order to get access to the links pointed to by these documentation files. Enter the activation code provided to you by your supplier with the product tape the next time you use `STRWWW` to start the server.

The activation code dictates which installed I/NET products can be used. As such, the code provided to you must reflect every product you intend to run when starting the servers. Please contact your supplier if you did not receive an activation code or your activation code does not work.

1.3 Viewing Merchant/400 Documentation from a Browser

All of the documentation is written in **HTML** (Hyper Text Markup Language) which will allow you to view it through a Web browser in one of two ways:

1.3.1 Accessing Document Files Directly

Connect to the AS/400 through a network drive and use a web browser that is written to work with Microsoft Windows or OS/2.

Upon installation, the Merchant/400 user guide documentation is located in the directory `/WWWServ/WebDocs/Shipped/Pubs/Merch400`. Most web browsers have the ability to load and view **HTML** files directly from a disk. Therefore, if your browser has this ability you can load the files through this means.

The Client Access/400 for Windows 3.1, Client Access/400 for Windows 95, or Client Access/400 Optimized for OS/2 software packages allow you to connect to the IFS Root file system through a network drive definition. In order to reach the necessary drive, you can either connect a drive directly to the `/WWWServ/WebDocs/Shipped/Pubs/Merch400` directory or you can connect to the ROOT of the IFS file system and work your way through the path (`/WWWServ/WebDocs/Shipped/Pubs/Merch400`) to reach the documentation.

Since the documentation is an **HTML** marked up document, there are many files which make up the entire documentation. The base document is named `usrguide.htm`. Access this file to start at the beginning of the document.

1.3.2 Accessing Documentation through Web Server/400 and Commerce Server/400

Once the server is started, the documentation is available through the server. Upon installation, the Merchant/400 user guide documentation is located in the directory `/WWWServ/WebDocs/Shipped/Pubs/Merch400`. If the server is started with the shipped configuration the documentation can be accessed through the following URLs:

`/Shipped/Pubs/Merch400/` or `/Shipped/Pubs/Merch400/usrguide.htm`

2. Getting Setup as an Internet Merchant

2.1 Getting Set Up as an Internet Merchant

2.1.1 So, You Want to do Business on the Internet

Transacting business over the Internet can be straightforward and efficient, but you need to set up accounts with partner processing agencies to make it all work. As an Internet merchant, you have a number of options available to you as you approach these partner alliances. Which options you select depend upon the links you wish to make when communicating with the processing center or acquirer (section 2.2 on page 12) while executing a credit card transaction.

2.1.2 Partner Host Companies That Provide the Link

Credit Card Network (section 2.3 on page 14) provides processing services for credit card transactions, both in an automated fashion via Merchant/400 and Holonet IAT, and through a manual entry program for special case transactions. Credit Card Network can also provide "one stop" registration to create your Merchant account with Credit Card Network, Holonet IAT, and a credit card processor.

Anacom Communications, Inc. (section 2.5 on page 16) allows for simple, low cost credit card processing. Anacom also supports merchants who do Internet business outside of the United States. Anacom also has a referral service to assist you in setting up a merchant account with a card processor.

Holonet IAT (section 2.4 on page 15) specializes in Internet communications and may be used to link to Credit Card Network, Anacom Communications, or other services. If desired, Holonet can assist with the sequential submission of transactions to multiple services when it is difficult to complete a transaction because of network or system failures. For example, Holonet may automatically submit a transaction to Anacom Communications if they are unable to connect to Credit Card Network to complete it.

DataCash Ltd. (section 2.6 on page 17) allows for simple, low cost credit card processing. DataCash specializes in supporting merchants who do Internet business outside of the United States and who process transactions in multiple currencies.

Each of these companies provides varying products that can assist you, the merchant, when processing a credit card transaction over the Internet. Each of them provides secure connections, using the SSL encryption method for all transmissions originated from Merchant/400. This is also not intended to be an all-inclusive list and I/NET does not recommend any of these companies over the others, nor over other agencies that are not yet included here. The choice of a partner agency is dependent on the features and options you wish to employ as an Internet Merchant. The topic How It All Fits Together (section 2.9 on page 20) can assist you in the steps to create the accounts necessary to do business on the Internet.

2.2 The Processing Center and Your Bank

2.2.1 The Beginning of Becoming an Internet Merchant

The process of selecting a credit card processor (sometimes referred to as a card acquirer or acquiring bank or institution) usually begins with contacting your financial institution. It should be able to provide you with the names of two or three providers with which they are familiar and have an established business relationship. If they are unable to provide you with the names of providers, a large regional financial institution in your area should be able to do so.

The actual selection should begin with your financial institution's referral. Remember that the relationship between the financial institution and the credit processor is critical to your receiving prompt and accurate deposits and detailed transaction reporting. Since this operation goes on behind the scenes, it is critical that there is no reason for your customers to feel uncomfortable with using the Internet to transact business. This behind the scenes processing must be seamless and unobtrusive to your customers.

Your selection of a processor should be made with the same criteria you use to establish any professional relationship. It must "feel right," as you will be working closely with the organization in establishing the processing procedures. After the process has been established you will probably not have very much interaction with this organization except for transaction reporting.

2.2.2 What You Will Need to Register With a Card Processor

The credit card processing company will need background information about your company, such as:

- The business's legal name
- Type of organization, i.e., corporation, etc.
- Bank relationship information
- How long the organization has been in business
- Whether your premises are owned or leased
- Credit references
- Credit card refund policies
- Type of products you will be selling by credit card
- Ownership information

They will also require a review of other background information such as recent financial statements, your marketing plan, and possibly your website (if you have one). They may insist on visiting your place of business and reviewing your business environment.

You can expect to pay an application fee to process your application. Once it is approved you can anticipate doing business immediately.

Credit card processing fees vary by the volume of business you transact and the dollar amount of these transactions. You will pay a set percentage of each sale in addition to an amount per transaction. You can elect to receive weekly or monthly statements that detail the history of your activity.

Part of the process of "signing up" with a partner processing agency will be to provide them the terminal identifier (section 2.7 on page 18) for your "virtual" terminal that has been assigned by the card processor. It is important to request this identifier when working with the card processor and keep it with your processor account information for reference when contacting partner agencies.

2.3 Credit Card Network

2.3.1 A Credit Card Transaction House for the World Wide Web

Credit Card Network is affiliated with many credit card processing centers, making it easy for you, the merchant, to do credit card authorization and transaction consummation over the Internet.

You can get an account set up with Credit Card Network by completing the secure Sign-Up form (https://www.creditnet.com/merchant/ccn-inet.html) for I/NET customers at their web site or by requesting information via their Response form (https://www.creditnet.com/merchant/ccncall3.html). You will need to have information about your company, as well as your bank and credit card processor, available for entry in the form. You will also want to have some ideas for a username (merchant name) and password.

Once you have a merchant account set up, Credit Card Network has support available for you to process your credit card transactions. This can be done via the Merchant/400 API or the secure Credit Card Network Manual processing screen.

2.3.2 Credit Card Network Advertised Features and Benefits

- On-line realtime credit card authorizations. Processing takes less than 45 seconds in most cases
- Accepts many of the major credit cards
- Low transaction fee which is in addition to the bank interchange rate.
- Money in your checking account in 48 hours
- Manual processing screen for other orders and transactions
- Daily transaction report for each merchant in password protected directory
- No minimum sale. No charge to customers/cardholders. No charge for denials.
- Verifone terminal not required
- ICVerify software not required
- CyberCash not required

For questions or comments about Credit Card Network, call (206)287-1794, write to ccnetwork@creditnet.com (mailto:ccnetwork@creditnet.com), or visit their web page (http://www.creditnet.com).

2.4 Holonet IAT

2.4.1 Transparent Processing for Credit Card Transactions

Holonet IAT is directly associated with Credit Card Network, Anacom Communications, and other credit card processing companies. This gives you, the merchant, the flexibility to process your transactions with one of many different organizations, depending on the circumstances of the transaction and the parameters set by your customer. Transmissions to Holonet via the Merchant/400 API are automatically routed, based on parameters you supply and your Holonet user account, to the proper destination with the transaction information securely delivered for processing. And it is all completed over the Internet.

There are a couple of different ways to set up an account with Holonet IAT. Simply creating a "transparent processing" account with Credit Card Network (section 2.3 on page 14) is one. By specifying that you wish to work through Holonet, an account will automatically be created for you with the same attributes as your Credit Card Network account. Holonet IAT is also an Internet Service Provider. If you wish to make your connection to the Internet directly via them, you can set up an account by calling Holonet at (510)704-0160, or you can get additional information at their Dedicated IP Access (http://www.holonet.net/holonet/dedicated.html) page.

Once you have an account set up, Holonet IAT will be almost transparent as you process your credit card transactions.

For questions or comments about Holonet IAT, call (510)704-0160, write to info@holonet.net (mailto:info@holonet.net), or visit their web page (http://www.holonet.net).

2.5 Anacom Communications, Inc.

2.5.1 Payment Solutions for Internet Merchants Worldwide

Anacom Communications, through their Secure PayTM program, promotes Internet Commerce by providing low cost order processing and payment solutions.

You can set up an account with Anacom by completing the secure Application form (https://www.anacom.com/forms/secforms/secure_f.htm?secure=securepayb&application=Merchant400) at their web site or by receiving information at their Credit Card Payment Services (http://www.anacom.com/payment.htm) page. You will need to have information about your company, as well as your bank and credit card processor, available for entry in the form. You will also want to have some ideas for a username (merchant name) and password. Make sure to reference that you are a Merchant/400 user in the comments area of the application form as well.

They also provide a referral service to assist you in setting up a merchant account. You can find it at their Merchant Account Enrollment (http://www.anacom.com/merchant.htm) page.

Once you have a Merchant account set up, Anacom has support available for you to process your credit card transactions. This can be done via the Merchant/400 API or a secure Anacom Manual processing screen.

2.5.2 Anacom Communications Advertised Features and Benefits

- Visa, MasterCard, American Express, Discover, and other cards are accepted. No software "wallet," "e-cash" or membership needed as required by other services.
- Ability to charge or authorize / reserve amounts based on a "sale" or "book and ship" transaction type (Transaction Type on page 40) method.
- No need to change servers or your hosting service.
- Payment information is encrypted and secured through a link to Anacom’s commerce server from your web pages, providing confidence to your customers to pay on-line.
- E-mail containing the order information is sent to you when an order is placed, or can be retained on Anacom’s server for retrieval at your convenience.
- Referral resource available to assist you in arranging for a merchant account.

For questions or comments about Anacom Communications, call (972)304-5012, write to questions@anacom.com (mailto:questions@anacom.com), or visit their web page (http://www.anacom.com).

2.6 DataCash Ltd.

2.6.1 Multi-Currency Payment Solutions for Global Internet Merchants

The DataCash Transaction.raw plain-text process is a very simple interface to the DataCash clearing system. It is a proprietary gateway method that runs on the DataCash web server. Parameters are passed from the Merchant/400 API straight to the script without the intermediary of an HTML form. Results are returned in plain text with no formatting, ready to be parsed by Merchant/400 for return to the Merchant's application.

You can set up an account with DataCash by completing the secure Application form (https://www.datacash.com/stronghold/datacash_application_form.htm) at their web site or by receiving information at their Application & Software (http://www.datacash.com/stronghold/application.htm) page. You will need to have information about your company, as well as your bank and the acquiring bank you have selected, available for entry in the form. You will also want to have some ideas about the currency requirements of your application. Make sure to select I/NET in the reseller field and reference that you are a Merchant/400 user in the comments area of the application form as well.

2.6.2 DataCash Advertised Features and Benefits

- Visa, MasterCard/EuroCard, American Express, JCB, Switch, Solo, and other cards are accepted.
- Ability to charge or pre-authorize / reserve amounts based on a "sale" or "book and ship" transaction type (Transaction Type on page 43) method.
- Officially approved by and permanently connected to many Acquiring Banks, including NatWest Streamline, Girobank, American Express, Barclays Merchant Services, Royal Bank of Scotland. All transactions are authorized, confirmed or rejected in real time.
- DataCash is a Multi-Currency system. You are not only able to trade in more than 150 currencies, you are also able to have the funds credited to your account in one of 17 currencies.

For questions or comments about DataCash, call (in the UK) +44 171 820 7733, write to info@datacash.com (mailto:info@datacash.com), or visit their web page (http://www.datacash.com).

2.7 The Virtual Credit Card Terminal

2.7.1 The Connection to the Card Processor

You are probably already familiar with a credit card terminal - the terminal or "swipe" machine at your local restaurant, market, or hotel. This machine is a direct connection to your card processor and contains an internal serial number that uniquely identifies the terminal to the card processor when a connection is made. This identifier links the terminal, via the processor's records, to your processor account.

When working in a "virtual" or computer-to-computer environment, some form of unique identification must accompany any transaction that is sent to the card processor. This, again, is how the processor knows where the transaction originated and that it is being executed for your account. You will receive other identification information from the card processor (section 2.2 on page 12) when you set up your account. These "client numbers" help to identify you to the processor, but are not the same as the terminal identification number.

The reason for having both is that the client identification is usually unique to a customer and credit card type - hence, you may receive multiple client numbers, one for each credit card type that you wish to accept. At the same time, you may have more than one card terminal - one at your front desk, one in the back office, and the new virtual terminal you are creating for Internet processing. Each of these will also have a unique identifier.

Part of the process of "signing up" with a partner processing agency will be to provide them the terminal identifier for your "virtual" terminal that was assigned by the card processor. It is important to request and keep this identifier with your processor account information.

2.8 Your Merchant ID and Password

2.8.1 Your Keys to Internet Merchant Services

Now that you have created accounts with a credit card processor (section 2.2 on page 12) and partner processing agencies to link you to the card processor, you have chosen or received a merchant ID, a merchant name, and a merchant password. These are the links you need to identify your transaction and the accounts you wish to use for a Merchant/400 transmission.

The merchant ID was assigned by the credit card processor and is the link to your account there. You may have received multiple merchant IDs, each relative to the credit cards that you will accept and process. This ID is used internally by the credit card processor and is not needed by the Merchant/400 API to execute a credit card transaction. The virtual terminal identification (section 2.7 on page 18), received from the card processor, will be used by the partner processing agency to link your merchant name with the merchant ID.

The merchant name and password were assigned by your chosen partner processing agency. These are used to log on to their service to pass credit card transactions to the card processor. These are unique to your organization and should be kept in confidence, as they can be used not only to complete a credit card transaction, but also to make debit adjustments to your merchant account. Both entries are SSL encrypted (with the customer order and credit card information) when transmitted to the partner processing agency.

2.9 How It All Fits Together

2.9.1 The Steps to Take to Become an Internet Merchant

1. Get set up with a bank - you must have a savings or checking account.

2. Contact a Credit Card Processor (section 2.2 on page 12). Receive a Merchant Identification number and a virtual terminal ID to process transactions electronically.

3. Sign up with Credit Card Network (section 2.3 on page 14), Anacom Communications (section 2.5 on page 16), DataCash (section 2.6 on page 17), or another transaction processing company. Register a Merchant ID, username and password (section 2.8 on page 19) for access. They will also need to know the virtual terminal ID (section 2.7 on page 18) assigned by your card processor.

4. If necessary, register with Holonet IAT (section 2.4 on page 15) to pass transactions to the appropriate transaction processing company. This may also involve a specialized configuration for sequencing or other circumstances.

5. Program (section 3.1 on page 32) your application to use the Merchant/400 API. You may wish to use the I/NET store sample application (section 4.1 on page 60) as a beginning to your own store application.

6. Set up, connect and configure the connection to your Internet Service Provider (ISP) to communicate over the Internet. This may already have been completed to run Web Server/400 or Commerce Server/400.

7. Call the Merchant/400 API to test and run the authorization routines through the entire system path.

Merchant/400 can also be used with the Net.Commerce for AS/400 (section 2.10 on page 21) product (5798-NC2) from IBM to process credit card transactions. This interface provides both mall/store functionality and credit card processing without the need for extensive CGI programming.

2.10 Using Merchant/400 with Net.Commerce

2.10.1 Introduction

Merchant/400, being an independent credit card processing API, can be used with your own applications or your own CGI store. Special objects are included with Merchant/400 that allow you to use it with the Net.Commerce for AS/400 product (5798-NC2, 5798-NC3) from IBM. These objects replace the default behavior "do_payment" API functionality that is performed by Net.Commerce when the shopper is ready to pay for purchases and elects to process their order.

Information about Net.Commerce can be found at www.ibm.com/net.commerce (http://www.ibm.com/net.commerce/).

When installed, the Merchant/400 sample store (section 4.1 on page 60) is configured to call a test processing center at I/NET. As a result, when you put the Merchant/400 do_payment interface into place, you can test the credit card transaction process to insure that everything has been moved and updated correctly without actually incurring charges or fees. The test processing center will do minimal validation of the data passed to it and will either return a success or failure. The test processing center only accepts one test credit card. The credit card is: Master Card, 5499740000000032, and expires 1/2001. This is not a real credit card.

2.10.2 Functionality Provided by the Merchant/400 do_payment Interface

Once registered, special Merchant/400 objects perform tasks that retrieve information from the Net.Commerce process. They then read order, shopper and merchant data from Net.Commerce and Merchant/400 sample files, and process the credit card transaction. A record is also written to the MNCINFP Merchant/400 file to store approval information about the order.

2.10.2.1 AS/400 Objects Involved in the Interface

The following objects are included with the Merchant/400 product specifically to support the Net.Commerce do_payment interface:

`WWWMNCMA`
: A service program that enables the interface between Net.Commerce and Merchant/400. WWWMNCMA dynamically loads WWWMNCM2 and calls the NCApiDoPayment function found there. During the execution of WWWMNCMA, the library WWWSERVER is added to the job's library list. This is done to facilitate service program linking with the Merchant/400 API.

`WWWMNCM2`
: The service program that is the do_payment interface between Net.Commerce and Merchant/400. WWWMNCM2 is linked with the WWWMAPI Merchant/400 service program and contains the NCApiDoPayment functionality.

`err_do_payment.ncimrc.d2w`
: A Net.Data macro, located in the IFS, that delivers information to the shopper about errors that have occurred during the Merchant/400 do_payment process.

`orderdspp.ncimrc.d2w`
: A Net.Data macro, located in the IFS, that displays the detail of the shopper's order and allows them to insert credit card information to process payment for the order. Special codes have been added to include the shopper identification when the do_payment API is called as a result of this macro.

`orderacpt.ncimrc.d2w`
A Net.Data macro, located in the IFS, that provides the shopper with receipt information about their purchase, including the card processor approval code.

`NCSAMPLE.C`
An ILE C source member that allows you to customize the behavior of the Merchant/400 do_payment interface.

`NCSAMPL2.C`
An ILE C source member that allows you to customize the behavior of the Merchant/400 do_payment interface. This interface is particular to circumstances when non-registered shopper purchases are desired in a Net.Commerce version 3.2 environment.

`NVPAIR.C and NVPAIR.H`
ILE C source member and header information provided originally by IBM as part of the Net.Commerce Toolbox. These source members are compiled and used with NCSAMPLE.C to customize the do_payment interface.

`MNCINFP`
A file, shipped in the library WWWMRCH, that contains information about completed credit card transactions. This information includes the order number, credit card number, and approval code associated with the order. Care should be taken to secure this file as it contains private shopper information. The Net.Commerce database user (associated with your mall database instance) is probably the only user that needs access to this file.

`MNCINFL1`
A logical file, built over MNCINFP, in the WWWMRCH library that indexes the MNCINFP data by order ID. Any actions taken on MNCINFP (authority modifications, duplication, etc) should also be performed on this file.

`MGLOBALP`
The global configuration file that contains information about how the Merchant/400 sample store operates. This file can also be used by the do_payment interface to determine operational attributes, including remote processing host name, host port, and transmission method.

2.10.2.2 ***How the Interface Works***

The following steps are taken by the NCApiDoPayment functionality that is part of the interface:

1. The special orderdspp.ncimrc.d2w Net.Data macro is called to display the order detail and collect credit card information. Shopper identification information is also passed along to the WWWMNCMA program at this point.
2. Resolve to the do_payment interface service progam (WWWMNCM2) and to the NCApiDoPayment API function.
3. Execute the NCApiDoPayment function in WWWMNCM2.
4. Retrieve Name/Value Pair parameters from Net.Commerce. This includes the credit card type, number, expiration date, and shopper identification.
5. Write a record to the ORDPAYMTHD file to flag the order and identify the payment method (credit card information) for the order.
6. Read (via SQL) the ORDERS file to retrieve the external order number.
7. Read (via SQL) the SHADDR file to retrieve the shopper's name and address information for the Merchant/400 API call.
8. Read (via SQL) the MGLOBALP (Global Configuration Physical File on page 65) file to get the Merchant's login information for the Merchant/400 API call. Transmission method and transaction type information is also retrieved at this point.

9. Set up the parameters and structure values (section 3.5 on page 40) for the call to the Merchant/400 API.
10. Add the library WWWSERVER to the job's library list. This is done to allow for the linkage of Merchant/400 and Commerce Server/400 service programs when the Merchant/400 API is executed.
11. Execute (section 3.3 on page 35) the Merchant/400 WwwMrcSubmit API function.
12. Write a record to the MNCINFP (Net.Commerce Order Information Physical File on page 69) file to record the order information and the transaction approval code (Approval Code on page 45) - only if the transaction completed normally (approved).
13. Return the appropriate Net.Commerce response (via return codes and Name/Value Pair parameters) based on information from the Merchant/400 API.
14. Once the do_payment functionality has completed, Net.Commerce calls the appropriate Net.Data macro to present the results of the API to the shopper. If the transaction was approved, orderacpt.ncimrc.d2w is called. If an error occurred, the err_do_payment.ncimrc.d2w macro is called.
15. If orderacpt.ncimrc.d2w is called because of a successful transaction, it reads the MNCINFL1 file (via SQL) retrieving the record associated with the order being processed. It then displays receipt information, including the approval code from the file, to the shopper via HTML.
16. If err_do_payment.ncimrc.d2w is called because of an error, it retrieves the error information from the Name/Value Pairs parameter data, formats it, and displays it to the shopper for their use during problem determination. An opportunity is also provided for the shopper to "go back" to return to the previous screen to make corrections or modifications.

If a Book type transaction (Transaction Type on page 40) was placed, it is the merchant's responsibility to develop and execute the appropriate programming to call the Merchant/400 API to register the corresponding Ship transaction once the order is completed.

2.10.3 Configuring the Merchant/400 Objects for Use with Net.Commerce

To configure the special Merchant/400 objects for use in a Net.Commerce store environment you will need to move them to a new location and configure the Net.Commerce application to find them. You will also need to update the database to contain information that is appropriate to your online store environment.

The following is a list of steps that need to be performed to prepare the Merchant/400 objects for use with Net.Commerce:

1. Copy the Merchant/400 files needed, the Net.Data macros and prepare the newly copied objects for use with your new Net.Commerce mall. The following steps detail these functions:
   - Copy the Net.Data macros (`orderdspp.ncimrc.d2w`, `err_do_payment.ncimrc.d2w` and `ordracpt.ncimrc.d2w`). These are shipped as IFS stream files in `/WWWSERV/WEBDOCS/SHIPPED/MERCHANT`. The default location for the Net.Commerce macros is the directory `/QIBM/ProdData/NetCommerce/macro/Mri2924/Ncsample` and you may copy them there. If the macro directory for your Net.Commerce mall instance is not `/QIBM/ProdData/NetCommerce/macro/Mri2924/Ncsample` then you should copy the files to the appropriate macro directory for your server. If another version of Merchant/400 is installed, the objects in `/WWWSERV/WEBDOCS/SHIPPED/MERCHANT` will be overwritten. The copy can be done using the `CPY` command. For example, to copy the `orderdspp.ncimrc.d2w file`, you could enter:
     ```
     CPY OBJ('/WWWServ/WebDocs/Shipped/Merchant/orderdspp.ncimrc.d2w')
     TODIR('/QIBM/ProdData/NetCommerce/macro/Mri2924/Ncsample')
     ```
   - Copy the `MGLOBALP`, `MNCINFP` and `MNCINFL1` files from `WWWMRCH` to your mall database library that was created when you configured Net.Commerce. This will segregate your Merchant/400 mall information from other Merchant/400 sample data. In addition, if another version of Merchant/400 is installed, the objects in `WWWMRCH` will be overwritten. The copy

can be done using the CRTDUPOBJ command. Don't forget to copy the physical files (MGLOBALP and MNCINFP) first.

- Start journalling (required by the Net.Commerce mall application) on the physical file MNCINFP. The command to do this is:

```
STRJRNPF FILE(library/MNCINFP) JRN(library/QSQJRN)
IMAGES(*BOTH) OMTJRNE(*OPNCLO)
```

Note that the CPY and CRTDUPOBJ commands will create the new objects with you as the owner and WWWUSER with *ALL object authority. This authority does not include your Net.Commerce mall database instance user (based on the collection assigned when you created the mall in Net.Commerce), so the application may not function properly. You will need to provide the read, write and execute authority needed for your mall user. This can be done using the GRTOBJAUT and CHGAUT commands. You may also wish to change the owner of these objects back to WWWUSER or to the user associated with your Net.Commerce mall.

2. Update the global configuration physical file (MGLOBALP) (Global Configuration Physical File on page 65) in your mall library. This physical file contains a single record that stores a variety of information that the application needs to run. This is also where you set many of the Merchant/400 API parameters, like host service name, transmission method, partner host name, and partner SSL port, that will be used for your credit card transactions. Update the file (using DFU or another utility of your choosing) to contain the parameter information that relate to your application and environment.
3. Register and activate the new objects in the Net.Commerce databases. Please refer to the instructions that relate specifically to Net.Commerce version 2 (section 2.11 on page 26) and Net.Commerce version 3.2 (section 2.12 on page 28) for assistance with this task. Don't forget to change your browser cache and proxy options. Also, the browser *Back* button does not work very well when using the Net.Commerce Administrator pages.

2.10.4 Creating Your Own API - NCSAMPLE.C and NCSAMPLE2.C

Source is provided for you to create your own Net.Commerce - Merchant/400 interfaces. The ILE C member NCSAMPLE can be found in the file WWWSERVER/QCSRC, as well as NVPAIR. An ILE C header file member, NVPAIR, can also be found in the file WWWSERVER/H. These are provided as sample code to assist you in the understanding of the code requirements for do_payment functionality during the development of your own API.

NCSAMPLE is slightly different from the WWWMNCMA and WWWMNCM2 functionality in that it performs the SQL database operations and then dynamically loads the WWWMAPI service program and executes the WwwMrcSubmit function. WWWMNCMA dynamically loads the WWWMNCM2 service program, which does the database operations and executes WwwMrcSubmit directly (WWWMAPI was linked when WWWMNCM2 was created). NCSAMPLE was created in this manner so you only have one source member to modify and compile when creating your unique API functionality.

NCSAMPLE may be modified as needed to accomodate your individual merchant environment. Specifications for the CRTSQLCI and CRTSRVPGM commands are included in special build notes that can be found at the beginning of the NCSAMPLE source. The NVPAIR source (both ILE C and header) were originally provided by IBM as part of the Net.Commerce Toolbox. As such, it is not recommended that modify either of these files. Information in the Net.Commerce documentation will also be helpful in creating the module and service program objects for your API. Your NCSAMPLE service program may be registered for use by the Net.Commerce in the same manner as the Merchant/400 customized API, described earlier (Register and activate on page 24).

The Net.Data macros (orderdspp.ncimrc.d2w, err_do_payment.ncimrc.d2w and ordracpt.ncimrc.d2w) that are shipped as IFS stream files in /WWWSERV/WEBDOCS/SHIPPED/MERCHANT may also be tailored to meet your individual needs to present the appropriate interfaces to the shoppers of your mall. If another version of Merchant/400 is

installed, the objects in `/WWWSERV/WEBDOCS/SHIPPED/MERCHANT` will be overwritten, so it is recommended that you make modifications for your mall in a different directory. Your order display, approval and error macros may be registered for use by the Net.Commerce in the same manner as the Merchant/400 customized macros, described earlier (Register and activate on page 24).

In addition, for those circumstances when non-registered shopper purchases are desired in a Net.Commerce version 3.2 environment, `NCSAMPL2` and `orderdspp2.ncimrc.d2w` are also provided as samples to collect credit card name and address information that is needed during the purchase process. These may also be duplicated and modified to suit the individual needs of the online mall or store.

2.11 Configuring Merchant/400 Objects for Use with Net.Commerce version 2

To enable the special Merchant/400 objects for use in a Net.Commerce version 2 store environment, you will need to configure the Net.Commerce application to register the customized API and macros so it can find and use them. To do this, you will need to execute various Net.Commerce configuration routines to reflect the new names and references. As a result, some knowledge of the Net.Commerce browser-based configuration will be helpful for the completion of this task.

2.11.1 **Activating the Customized do_payment API in Net.Commerce version 2**

Perform the following steps to activate the customized do_payment API functionality that will execute credit card transactions via Merchant/400 (please refer to the Net.Commerce documentation for additional information about the assignment of API tasks and macros):

1. Ensure that you are authorized to replace the DO_PAYMENT task.
2. Open the Net.Commerce Administrator using your browser.
3. On the task bar, click Site Manager.
4. Click Task Management to display the Task Management form.
5. Select the API task type from the Select Task Type drop-down list. A list of tasks appears in the bottom frame.
6. Select the DO_PAYMENT task name from the list. The fields automatically fill with task information.
7. Go to the API DLL/Function Assignment form by clicking Task Assignment and then API DLL/Function.
8. Enter the following in the API DLL Name and API Function Name fields on the API DLL/Function Assignment form:

   The API service program name and the API service program function name.
   For the service program name, use the format: <library>/<srvpgm_name>, or <srvpgm_name>. In the last case, the name will be resolved at runtime using the library list.
   The API service program function name is case-sensitive and should contain the text "NCApiDoPayment". As a result, for the objects shipped with Merchant/400:
   - In API DLL Name type the service program name: `WWWSERVER/WWWMNCMA` (or `WWWSERVER/NCSAMPLE` for your own customized API (Creating Your Own API - NCSAMPLE.C and NCSAMPLE2.C on page 24))
   - In API Function Name type the service program function name: `NCApiDoPayment`
9. Click Save.

2.11.2 **Activating the Customized Net.Data macros in Net.Commerce**

Special Net.Data macros are included to retrieve specialized data about the order and shopper, and to deliver Merchant/400 information to the shopper about the results of their credit card transaction. Once the macro files are moved into place, you should perform the following steps to activate the customized macro for do_payment errors:

1. Ensure that you are authorized to replace the DO_PAYMENT error macro.
2. Open the Net.Commerce Administrator using your browser.
3. On the task bar, click Site Manager.
4. Click Task Management to display the Task Management form.
5. Select the API task type from the Select Task Type drop-down list. A list of tasks appears in the bottom frame.
6. Select the DO_PAYMENT task name from the list. The fields automatically fill with task information.
7. Go to the Macro Assignment form by clicking Task Assignment and then API Error Macro....

8. Enter the following in the Macro Filename field on the Macro Assignment form:
   - The Merchant/400 Net.Data macro name `err_do_payment.ncimrc.d2w`, or
   - The macro name of a do_payment error Net.Data macro that you have tailored to your individual requirements.
9. Click Save.

Perform the following steps to activate the customized macro for order acceptance:

1. Ensure that you are authorized to replace the ORD_OK macro.
2. Open the Net.Commerce Administrator using your browser.
3. On the task bar, click Site Manager.
4. Click Task Management to display the Task Management form.
5. Select the ORDER task type from the Select Task Type drop-down list. A list of tasks appears in the bottom frame.
6. Select the ORD_OK task name from the list. The fields automatically fill with task information.
7. Go to the Macro Assignment form by clicking Task Assignment.
8. Enter the following in the Macro Filename field on the Macro Assignment form:
   - The Merchant/400 Net.Data macro name `orderacpt.ncimrc.d2w`, or
   - The macro name of an order acceptance Net.Data macro that you have tailored to your individual requirements.
9. Click Save.

Perform the following steps to activate the customized macro for order detail display:

1. Ensure that you are authorized to replace the ORD_DSP_PEN macro.
2. Open the Net.Commerce Administrator using your browser.
3. On the task bar, click Site Manager.
4. Click Task Management to display the Task Management form.
5. Select the ORDER task type from the Select Task Type drop-down list. A list of tasks appears in the bottom frame.
6. Select the ORD_DSP_PEN task name from the list. The fields automatically fill with task information.
7. Go to the Macro Assignment form by clicking Task Assignment.
8. Enter the following in the Macro Filename field on the Macro Assignment form:
   - The Merchant/400 Net.Data macro name `orderdspp.ncimrc.d2w`, or
   - The macro name of an order acceptance Net.Data macro that you have tailored to your individual requirements.
9. Click Save.

Please refer to the Net.Commerce version 2 documentation for additional information about the assignment of API tasks and macros.

2.12 Configuring Merchant/400 Objects for Use with Net.Commerce version 3.2

To enable the special Merchant/400 objects for use in a Net.Commerce version 3.2 store environment, you will need to configure the Net.Commerce application to register the customized API and macros so it can find and use them. To do this, you will need to modify various Net.Commerce database files to reflect the new names and references. As a result, some AS/400 database or SQL knowledge will be helpful for the completion of this task. Knowledge of the Net.Commerce browser-based configuration will also be helpful while activating the Net.Data macros. It is important to note that the Merchant/400 integration is accomplished via Process Task API Execution rather than Overridable Functions, since the functionality and configuration of each can be easily confused.

Special thanks to Martin Holmes of Morpheus, Ltd. (www.morpheus.ltd.uk) for his work on this page and the API registration technique that it employs.

2.12.1 **Activating the Customized do_payment API in Net.Commerce version 3.2**

Perform the following steps to activate the customized do_payment process task API functionality that will execute credit card transactions via Merchant/400 (please refer to the Net.Commerce documentation for additional information about the assignment of process task APIs and macros):

1. **Ensure that you are authorized to replace the DO_PAYMENT task.**

2. **Register the Merchant/400 Payment API in Net.Commerce.**

   - Insert a record into the **APIS** database table that resides in your Net.Commerce instance library. Example values are shown in the table below:

APIRFNBR	APIMERNBR	APIDLLNAME	APIFUNCNAME
2015	<merchant_id>	WWWSERVER/WWWMNCMA	NCApiDoPayment

NOTE: The <merchant_id> reference is the `MERFNBR` value for your store in the `MERCHANT` file in your instance library.

3. **Override the DO_PAYMENT process with the Merchant/400 API.**

   - Insert a record into the **TASK_MER_OF** table within your instance library. Example values are shown below:

TASK_RN	MERCHANT_RN	OF_RN	SEQUENCE
2015	<merchant_id>	2215	-

NOTE: The `OF_RN` reference is the `REFNUM` value for the DoPaymentProxy function in the OFS table in your instance library.

2.12.2 Activating the Customized Net.Data macros in Net.Commerce

Special Net.Data macros are included to retrieve specialized data about the order and shopper, and to deliver Merchant/400 information to the shopper about the results of their credit card transaction. Once the macro files are moved into place, you should perform the following steps to activate the customized macro for do_payment errors:

1. Ensure that you are authorized to replace the DO_PAYMENT error macro.
2. Open the Net.Commerce Administrator using your browser.
3. On the task bar, click Site Manager.
4. Click Task Management to display the Task Management form.
5. Select the API task type from the Select Task Type drop-down list. A list of tasks appears in the bottom frame.
6. Select the DO_PAYMENT task name from the list. The fields automatically fill with task information.
7. Go to the Macro Assignment form by clicking Task Assignment and then API Error Macro....
8. Enter the following in the Macro Filename field on the Macro Assignment form:
   - The Merchant/400 Net.Data macro name `err_do_payment.ncimrc.d2w`, or
   - The macro name of a do_payment error Net.Data macro that you have tailored to your individual requirements.
9. Click Save.

Perform the following steps to activate the customized macro for order acceptance:

1. Ensure that you are authorized to replace the ORD_OK macro.
2. Open the Net.Commerce Administrator using your browser.
3. On the task bar, click Site Manager.
4. Click Task Management to display the Task Management form.
5. Select the ORDER task type from the Select Task Type drop-down list. A list of tasks appears in the bottom frame.
6. Select the ORD_OK task name from the list. The fields automatically fill with task information.
7. Go to the Macro Assignment form by clicking Task Assignment.
8. Enter the following in the Macro Filename field on the Macro Assignment form:
   - The Merchant/400 Net.Data macro name `orderacpt.ncimrc.d2w`, or
   - The macro name of an order acceptance Net.Data macro that you have tailored to your individual requirements.
9. Click Save.

Perform the following steps to activate the customized macro for order detail display:

1. Ensure that you are authorized to replace the ORD_DSP_PEN macro.
2. Open the Net.Commerce Administrator using your browser.
3. On the task bar, click Site Manager.
4. Click Task Management to display the Task Management form.
5. Select the ORDER task type from the Select Task Type drop-down list. A list of tasks appears in the bottom frame.
6. Select the ORD_DSP_PEN task name from the list. The fields automatically fill with task information.
7. Go to the Macro Assignment form by clicking Task Assignment.
8. Enter the following in the Macro Filename field on the Macro Assignment form:
   - The Merchant/400 Net.Data macro name `orderdspp.ncimrc.d2w`, or
   - The macro name of an order acceptance Net.Data macro that you have tailored to your individual requirements.
9. Click Save.

Please refer to the Net.Commerce version 3.2 documentation for additional information about the assignment of API process tasks and macros.

3. Using Merchant/400

3.1 Using the Merchant/400 API

3.1.1 Introduction

Merchant/400 API interfaces are provided for two different programming environments:

- ILE (e.g., ILE C, ILE RPG, ILE COBOL)
- OPM (e.g., RPG, COBOL, CL)

An OPM interface is provided by a program in library WWWSERVER.

The WWWMAPI service program in library WWWSERVER provides the ILE entry point. A bind directory named MBNDDIR in library WWWSERVER is provided to aid in developing ILE applications.

3.1.2 Files Provided to Aid your Development Effort

Includes are provided to aid in developing applications in the following programming environments:

ILE C

Prototypes and structures are provided for ILE C applications. See the member WWWMAPI (/qsys/wwwserver/h/WWWMAPI.txt) in source file H (i.e., WWWMAPI.H) in library WWWSERVER for more information on ILE C interfaces. More information is also available in the topic that discusses calling the API from an ILE program (section 3.3 on page 35).

ILE RPG

Constants that define ILE API entry points for ILE RPG applications are available in member DWWWMAPI (/qsys/wwwserver/qrpglesrc/DWWWMAPI.txt) in WWWSERVER/QRPGLESRC. More information is also available in the topic that discusses calling the API from an ILE program (section 3.3 on page 35).

ILE COBOL

Data items that define ILE API entry points for ILE COBOL applications are available in member WWWMAPI (/qsys/wwwserver/qcbllesrc/WWWMAPI.txt) in WWWSERVER/QCBLLESRC. More information is also available in the topic that discusses calling the API from an ILE program (section 3.3 on page 35).

RPG

Data items that define API entry points for RPG/400 applications are available in member WWWMAPI (/qsys/wwwserver/qrpgsrc/WWWMAPI.txt) in WWWSERVER/QRPGSRC. More information is also available in the topic that discusses calling the API from an OPM program (section 3.4 on page 38).

COBOL

Data items that define API entry points for COBOL/400 applications are available in member WWWMAPI (/qsys/wwwserver/qcblsrc/WWWMAPI.txt) in WWWSERVER/QCBLSRC. More information is also available in the topic that discusses calling the API from an OPM program (section 3.4 on page 38).

The section that discusses error conditions (section 3.6 on page 50) should also be reviewed when writing programs to access the API.

3.2 What is a Merchant Server?

3.2.1 A Way to do Business Over the Internet

Do you know how to process a credit card? Many people think of the terminal or "swipe" machine at their local restaurant, market, or hotel. Most business applications require that you bill your customer or accept credit card information to be keyed manually before the order is filled. This can mean slower response to your customers, delayed order shipping, and special security requirements to keep your customer's credit card information safe. **Not so anymore!!!**

With Merchant/400, used in conjunction with Commerce Server/400, you can now call a simple API (Application Programming Interface) to process business transactions over the World Wide Web. You can use your AS/400 applications to collect credit card information from the customer while the order is being placed - making the transaction easier and more secure for your customer, resulting in new marketing, public relations and customer support opportunities.

3.2.2 The Process

Here's how it works. AS/400 applications collect information about an order - the customer, their credit card information, and the total amount to be charged. This is structured and sent via a call (section 3.1 on page 32) to the Merchant/400 API. Merchant/400 packages the information and sends it, via a secure SSL link, to a transaction processing center. Once received, the transaction center processes the transaction with your credit card processing center (section 2.2 on page 12). The results (approved or denied) are returned to your application - usually within 1 minute. And you can transact charges or pre-authorizations and shipments with Merchant/400 too, just by specifying a transaction type (Transaction Type on page 40).

Merchant/400 means your business doesn't need to rely on someone calling for credit card approval anymore. Your customers can make purchases and pay for them - immediately. No more waiting for the "credit authorization" person to get in. No more calls to inform the customer that their credit was denied and you can not fill their order. No more fumbling with the Veriphone terminal to enter a transaction.

In summary, Merchant/400 in conjunction with Commerce Server/400, gives you the following benefits:

- Quick authorization of charged amounts - at any time, day or night
- Ability to charge or authorize/reserve amounts, depending on your business needs
- Faster processing and delivery of customer orders
- Use of applications to collect information - no credit card terminal necessary
- More reliable completion of transactions - fewer "on account" sales

3.3 Using the Merchant/400 API in an ILE Program

3.3.1 Introduction

The Merchant/400 API interface provided for the ILE (Integrated Language Extensions) environment is through the service program WWWMAPI in the WWWSERVER library. Function prototypes and definitions are provided for ILE C applications in the member WWWMAPI (/qsys/wwwserver/h/WWWMAPI.txt) in source file H (i.e., WWWMAPI.H) in library WWWSERVER. Constants that define API entry points for ILE RPG applications are available in member DWWWMAPI (/qsys/wwwserver/qrpglesrc/DWWWMAPI.txt) in WWWSERVER/QRPGLESRC. Data items that define API entry points for ILE COBOL applications can be found in member WWWMAPI (/qsys/wwwserver/qcbllesrc/WWWMAPI.txt) in WWWSERVER/QCBLLESRC. A bind directory named MBNDDIR in library WWWSERVER is provided to aid in developing ILE applications.

3.3.2 The ILE Call to Execute the API

The following documents the function that is available to execute the Merchant/400 API in an ILE environment. The API requires all of the parameters that are listed. More information about the minimum information needed by the API can be found in the topic on Required Information (section 3.7 on page 53).

WwwMrcSubmit

ILE C Prototype:

```
void WwwMrcSubmit(char *pachRemoteHost,
                  long  lRemoteHostLen,
                  long  lRemotePort,
                  long  lTransmitMethod,
                  long  lInputFormat,
                  void *pSubmitInputs,
                  long  lOutputFormat,
                  void *pSubmitResults,
                  long *plResultSuccess);
```

where,

`pachRemoteHost` is the buffer containing the name and domain, or IP address, of the remote processing host. This will be supplied by the partner agency.

`lRemoteHostLen` is the length of the Remote Host Name buffer. This will be calculated by your application based on the size of the entry in the pachRemoteHost parameter.

`lRemotePort` is the socket port number used by the remote host for SSL communications. This will usually be set to `443`.

`lTransmitMethod` tells the Merchant/400 API how to transmit the data. Please refer to the topic What Happens - Transmission Methods (section 3.8 on page 55) for more information on how to set this parameter.

`lInputFormat` tells the API what format your input data (Input Formats and Structures. on page 40) will be in.

`pSubmitInputs` is the formatted buffer that holds the customer, order, and credit card information.

`lOutputFormat` tells the API what format you are expecting to receive results data (Output Formats and Structures. on page 46) to be in.

`pSubmitResults` is the formatted buffer that will hold the returned transaction information from the card processor.

`plResultSuccess` is the code returned (section 3.6 on page 50) indicating the success of the transaction.

ILE RPG or COBOL Parameters:

```
1.  Remote Host Name            Input        Char(*)
2.  Remote Host Name Length     Input        Binary(4)
3.  Remote Port Number          Input        Binary(4)
4.  Transmission Method         Input        Binary(4)
5.  Input Data Format           Input        Binary(4)
6.  Input Data Buffer           Input        Char(*)
7.  Output Data Format          Input        Binary(4)
8.  Output Data Buffer          Input/Output Char(*)
9.  Completion Indicator        Output       Binary(4)
```

where,

`Remote Host Name` is the buffer containing the name and domain, or IP address, of the remote processing host. This will be supplied by the partner agency.

`Remote Host Name Length` is the length of the Remote Host Name buffer. This will be calculated by your application based on the size of the entry in the Remote Host Name parameter.

`Remote Port Number` is the socket port number used by the remote host for SSL communications. This will usually be set to `443`.

`Transmission Method` tells the Merchant/400 API how to transmit the data. Please refer to the topic What Happens - Transmission Methods (section 3.8 on page 55) for more information on how to set this parameter.

`Input Data Format` tells the API what format your input data (Input Formats and Structures. on page 40) will be in.

`Input Data Buffer` is the formatted buffer that holds the customer, order, and credit card information.

`Output Data Format` tells the API what format you are expecting to receive results data (Output Formats and Structures. on page 46) to be in.

`Output Data Buffer` is the formatted buffer that will hold the returned transaction information from the card processor.

`Completion Indicator` is the code returned (section 3.6 on page 50) indicating the success of the transaction.

Please refer to the document What Happens - Transmission Methods (section 3.8 on page 55) for a description of the processing that is performed within the API.

3.4 Using the Merchant/400 API in an OPM Program

3.4.1 Introduction

The Merchant/400 API interface provided for the OPM (Original Programming Model) environment is through the program WWWMRCSB in the WWWSERVER library. Constants that define API entry points for OPM RPG applications are available in member WWWMAPI (/qsys/wwwserver/qrpgsrc/WWWMAPI.txt) in WWWSERVER/QRPGSRC. Data items that define API entry points for COBOL applications can be found in member WWWMAPI (/qsys/wwwserver/qcblsrc/WWWMAPI.txt) in WWWSERVER/QCBLSRC.

3.4.2 The OPM Program to Call to Execute the API

The following documents the program that is available to execute the Merchant/400 API in an OPM environment. The API requires all of the parameters that are listed. More information about the minimum information needed by the API can be found in the topic on Required Information (section 3.7 on page 53). The library `WWWSERVER` must be in your library list before calling the API.

WWWMRCSB

Parameters:

```
1.  Remote Host Name           Input        Char(*)
2.  Remote Host Name Length    Input        Binary(4)
3.  Remote Port Number         Input        Binary(4)
4.  Transmission Method        Input        Binary(4)
5.  Input Data Format          Input        Binary(4)
6.  Input Data Buffer          Input        Char(*)
7.  Output Data Format         Input        Binary(4)
8.  Output Data Buffer         Input/Output Char(*)
9.  Completion Indicator       Output       Binary(4)
```

where,

`Remote Host Name` is the buffer containing the name and domain, or IP address, of the remote processing host. This will be supplied by the partner agency.

`Remote Host Name Length` is the length of the Remote Host Name buffer. This will be calculated by your application based on the size of the entry in the Remote Host Name parameter.

`Remote Port Number` is the socket port number used by the remote host for SSL communications. This will usually be set to `443`.

`Transmission Method` tells the Merchant/400 API how to transmit the data. Please refer to the topic What Happens - Transmission Methods (section 3.8 on page 55) for more information on how to set this parameter.

`Input Data Format` tells the API what format your input data (Input Formats and Structures. on page 40) will be in.

`Input Data Buffer` is the formatted buffer that holds the customer, order, and credit card information.

`Output Data Format` tells the API what format you are expecting to receive results data (Output Formats and Structures. on page 46) to be in.

`Output Data Buffer` is the formatted buffer that will hold the returned transaction information from the card processor.

`Completion Indicator` is the code returned (section 3.6 on page 50) indicating the success of the transaction.

Please refer to the document What Happens - Transmission Methods (section 3.8 on page 55) for a description of the processing that is performed within the API.

3.5 Input and Output Formats

3.5.1 How to Exchange Data With the API

You, the merchant, have collected order information from the customer in your application program and are now ready to send it to the Merchant/400 API to approve the purchase and complete the order. This exchange of order and result information is done via formats and structures that have been created for each of the differing styles of data being transferred. The selected input and output structures are passed to the API as parameters with format identifiers to indicate to the API which structure to use for its internal processing of the passed data. The definitions of the structures and format identifiers may be found for ILE C programs in the `wwwmapi.h` include member. Definitions for other ILE and OPM programs can be found in the member `WWWMAPI` in their respective source files (QRPGSRC, QCBLSRC, and QCBLLESRC). ILE RPG data structure definitions may be found in the `DWWWMAPI` member. These definitions should be copied into your program source for use in your application.

3.5.1.1 Input Formats and Structures.

Customer, order, credit card, and other information is passed to the Merchant/400 API via the Input Data Buffer parameter and can take the form of the WWWMERCH_INPUT1, WWWMERCH_INPUT2 or WWWMERCH_INPUT999 structures. By specifying MERCH_INPUT1, MERCH_INPUT2 or MERCH_INPUT999 in the Input Data Format parameter, you inform the API that input data is in the WWWMERCH_INPUT1, WWWMERCH_INPUT2 or WWWMERCH_INPUT999 style structures, respectively. The data required (section 3.7 on page 53) by the host service and the transmission method (section 3.8 on page 55) desired will help determine which of these formats to choose for each transaction processed.

The WWWMERCH_INPUT1 structure has the following field elements (Note - The following field elements are formatted using HTML table definitions that are best viewed with a browser that supports tables):

Field Name	DataType	Length	Description
Host Service Name	binary	9,0	This entry is used to further determine the route processing for the transaction. If the transaction is to be processed through Holonet IAT, you must specify the type of service you wish to utilize. You currently have three choices: 1. MERCH_INPUT1_HOLOPAY - Use the direct HoloPay service 2. MERCH_INPUT1_CCN - Use the Credit Card Network service 3. MERCH_INPUT1_ANACOM - Use the Anacom Communications service If your transaction will not be routed through Holonet, this field is ignored and can contain any of the valid choices.
Transaction Type	binary	9,0	Merchant/400 includes the ability to charge an amount to a credit card via a "sale" transaction, and to request authorization for a charged amount, via a "book" transaction, reserving that amount to be charged later by a "ship" transaction. This entry is used to determine the process that will take place (sale, book, ship) for

			this transaction. If the transaction is to be processed through Anacom Communications, you must specify the type of transaction that this is. You currently have three choices: 1. MERCH_TRTYPE1_SALE - This is an immediate sale transaction. The amount specified in Order / Charge Amount (Order / Charge Amount on page 45) will be charged to the credit card. 2. MERCH_TRTYPE1_BOOK - This transaction will authorize the amount specified in Order / Charge Amount and reserve it against the credit limit associated with the card. The authorization and reserve is valid for a thirty (30) day period before it is automatically removed by the card processor. A "ship" transaction must be sent with the Result Code, and possibly Host Reference, from this transaction to actually charge an amount to the credit card. 3. MERCH_TRTYPE1_SHIP - This transaction will complete a previously issued "book" transaction. During this transaction, the authorization and reserve are modified to match the Order / Charge Amount specified and actually charged to the credit card. When this transaction is successfully completed, the authorization and reserve are removed and the Order / Charge Amount is charged to the credit card. Approval Code (Approval Code on page 45) is required for this type of transaction. If your transaction will not be routed through Anacom or DataCash, this field is ignored and can contain any of the valid choices.
Merchant Name	character	256	This is the merchant user name used during logon for transaction transfer. It is assigned by the the Internet partner agency when you created your merchant account. This is not the Merchant identification number assigned by your card processor.
Merchant Password	character	256	This is the merchant password used during logon for transaction transfer. It is assigned by the the Internet partner agency when you created your merchant account and is associated with your merchant name.
Order ID	character	64	A string that uniquely identifies the transaction. This is usually an order number, but can be any set of characters that allows you to track the transaction through each processing service and to your card processor.
Customer Name	character	64	The name of the customer for this transaction. This name must match the name found on the credit card.
Customer	character	128	This is the billing address for the customer card

Address			account. It must match the billing address for the credit card that is used for this transaction.
Customer City	character	64	Along with the Customer Address, this is the associated bill-to city for the customer card account.
Customer State/Province	character	64	The bill-to state or province associated with the customer card account.
Customer Zip/Postal Code	character	64	The bill-to zip code or postal code associated with the customer card account.
Customer Country	character	64	The bill-to country associated with the customer card account.
Credit Card Number	character	64	This is the number for the credit card being used for this transaction. You should not insert hyphens (-) or spaces at any points in the number.
Card Expiration Month	packed decimal	3,0	This is the month portion of the credit card expiration date. It is formatted in a "MM" format and must be a number from 1 to 12.
Card Expiration Year	packed decimal	5,0	This is the year portion of the credit card expiration date. It is formatted in a "YYYY" format. For example, a credit card that expires in July, 1998 would have a card expiration month of 07 and a card expiration year of 1998.
Order / Charge Amount	packed decimal	15,5	This is the amount of the order that is to be charged or authorized to the credit card identified in the credit card number field. This amount is to be valued in the currency of the merchant account at the card processing organization. For example, if your merchant account is set in United States dollars, this amount must be in US dollars. If you typically do business in another currency, you must calculate the conversion amount based on the currency of your merchant account.
Approval Code	character	256	This entry, associated with the MERCH_TRTYPE1_SHIP Transaction Type (Transaction Type on page 40), links this MERCH_TRTYPE1_SHIP transaction to a previous related "book" transaction. The value of this entry emanated from the credit card processor during the original "book" transaction and must match to complete the actual charge transaction.

The WWWMERCH_INPUT2 structure has the following field elements (Note - The following field elements are formatted using HTML table definitions that are best viewed with a browser that supports tables):

Field Name	DataType	Length	Description
Host Service Name	binary	9,0	This entry is used to further determine the route processing for the transaction. If the transaction is to be processed through Holonet IAT, you must specify the type of service you wish to utilize. You currently have three choices: 1. MERCH_INPUT1_HOLOPAY - Use the

			direct HoloPay service 2. MERCH_INPUT1_CCN - Use the Credit Card Network service 3. MERCH_INPUT1_ANACOM - Use the Anacom Communications service 4. MERCH_INPUT1_DATACASH - Use the DataCash service If your transaction will not be routed through Holonet, this field is ignored and can contain any of the valid choices.
Transaction Type	binary	9,0	Merchant/400 includes the ability to charge an amount to a credit card via a "sale" transaction, and to request authorization for a charged amount, via a "book" transaction, reserving that amount to be charged later by a "ship" transaction. This entry is used to determine the process that will take place (sale, book, ship) for this transaction. If the transaction is to be processed through Anacom Communications or DataCash, you must specify the type of transaction that this is. You currently have three choices: 1. MERCH_TRTYPE1_SALE - This is an immediate sale transaction. The amount specified in Order / Charge Amount (Order / Charge Amount on page 45) will be charged to the credit card. 2. MERCH_TRTYPE1_BOOK - This transaction will authorize the amount specified in Order / Charge Amount and reserve it against the credit limit associated with the card. The authorization and reserve is valid for a thirty (30) day period before it is automatically removed by the card processor. A "ship" transaction must be sent with the Result Code, and possibly Host Reference, from this transaction to actually charge an amount to the credit card. 3. MERCH_TRTYPE1_SHIP - This transaction will complete a previously issued "book" transaction. During this transaction, the authorization and reserve are modified to match the Order / Charge Amount specified and actually charged to the credit card. When this transaction is successfully completed, the authorization and reserve are removed and the Order / Charge Amount is charged to the credit card. Approval Code (Approval Code on page 45) is required for this type of transaction. Order ID (Order ID on page 44) will also be used to contain the previously received Host Reference Code (Result Host Reference on page 48) when issuing the "ship" via the DataCash service.

			If your transaction will not be routed through Anacom or DataCash, this field is ignored and can contain any of the valid choices.
Merchant Name	character	256	This is the merchant user name used during logon for transaction transfer. It is assigned by the the Internet partner agency when you created your merchant account. This is not the Merchant identification number assigned by your card processor.
Merchant Password	character	256	This is the merchant password used during logon for transaction transfer. It is assigned by the the Internet partner agency when you created your merchant account and is associated with your merchant name.
Order ID	character	64	A string that uniquely identifies the transaction. This is usually an order number, but can be any set of characters that allows you to track the transaction through each processing service and to your card processor. If the book or sale transaction is to be processed via DataCash, this must contain a numeric value between 6 and 9 digits long. In the case of a MERCH_TRTYPE1_SHIP (Transaction Type on page 43) transaction processed via the DataCash service, Order ID will be used to contain the previously received Host Reference Code (Result Host Reference on page 48) from the "book" transaction.
Customer Name	character	64	The name of the customer for this transaction. This name must match the name found on the credit card.
Customer Address	character	128	This is the billing address for the customer card account. It must match the billing address for the credit card that is used for this transaction.
Customer City	character	64	Along with the Customer Address, this is the associated bill-to city for the customer card account.
Customer State/Province	character	64	The bill-to state or province associated with the customer card account.
Customer Zip/Postal Code	character	64	The bill-to zip code or postal code associated with the customer card account.
Customer Country	character	64	The bill-to country associated with the customer card account.
Credit Card Number	character	64	This is the number for the credit card being used for this transaction. You should not insert hyphens (-) or spaces at any points in the number.
Card Expiration Month	packed decimal	3,0	This is the month portion of the credit card expiration date. It is formatted in a "MM" format and must be a number from 1 to 12.
Card Expiration Year	packed decimal	5,0	This is the year portion of the credit card expiration date. It is formatted in a "YYYY" format. For example, a credit card that expires in July, 1998 would have a card expiration month of 07 and a card expiration year of 1998.
Card Start Month / Issue Code	packed decimal	3,0	This is the month portion of the credit card start date, or the two-digit issue code number for the card. It is formatted in a "MM" or "nn" format and must be a number from 1 to 12 (for start date) or 1 to 99 (for

			issue code).
Card Start Year	packed decimal	5,0	This is the year portion of the credit card start date. It is formatted in a "YYYY" format. For example, a credit card that becomes active in July, 1998 would have a card start month of 07 and a card start year of 1998. In the case of using a card issue number, the start year should be passed with a value of zero (0000).
Order / Charge Amount	packed decimal	15,5	This is the amount of the order that is to be charged or authorized to the credit card identified in the credit card number field. This amount is to be valued in the currency of the merchant account at the card processing organization or in the currency indicated in the Currency (Currency of Charge Amount on page 45) field. For example, if your merchant account is set in United States dollars only, this amount must be in US dollars. If, on the other hand, you typically do business in other (or varied) currencies, you must use the Currency field and this amount must reflect the currency specified.
Approval Code	character	256	This entry, associated with the MERCH_TRTYPE1_SHIP Transaction Type (Transaction Type on page 43), links this MERCH_TRTYPE1_SHIP transaction to a previous related "book" transaction. The value of this entry emanated from the credit card processor during the original "book" transaction and must match to complete the actual charge transaction.
Currency of Charge Amount	character	64	This entry, associated with Order / Charge Amount (Order / Charge Amount on page 45), indicates the currency or denomination of the amount reflected. This should match the associated codes dictated by the Host Service (Host Service Name on page 42) used for this transaction. You should reference their documentation to determine the valid currencies supported and their associated codes.

The WWWMERCH_INPUT999 structure has the following field elements (Note - The following field elements are formatted using HTML table definitions that are best viewed with a browser that supports tables):

Field Name	DataType	Length	Description
Bytes Provided	binary	9,0	The numeric value of this field represents the amount of space that you have allocated and are prepared to send to the Merchant/400 API. It must be at least the size of the WWWMERCH_INPUT999 structure and should include any additional space needed for the transmit data in the submit input buffer.
Input Body CCSID	binary	9,0	This entry is used to inform the Merchant/400 API how the upline host intends to view the data you are transmitting in the portions of the submit input buffer. This is an ASCII CCSID that the Merchant/400 API will convert the body information to before transmitting it to the host.

			A CCSID of 819 is typically used for standard ASCII transmissions in US English.
Results Body CCSID	binary	9,0	This entry is used to inform the Merchant/400 API how the upline host intends to send the data you are expecting to receive in the portions of the Return Buffer (found in the WWWMERCH_RESULT999 output structure). This is an ASCII CCSID that the Merchant/400 API will convert the body information from after receiving it from the host and before returning it to your application. A CCSID of 819 is typically used for standard ASCII transmissions in US English.
Submit Input Buffer	character	????	The Submit Input Buffer field is not really part of the WWWMERCH_INPUT999 structure, but an area that is made available for the transmit data to be sent to the upline host system. This area will contain the HTML form data that you intend to transmit. The data received in this buffer should be in the job's CCSID and will be converted appropriately for transmission.

3.5.1.2 ***Output Formats and Structures.***

The results of the transaction (success or failure) are received from the Merchant/400 API via the Output Data Buffer parameter and can take the form of the WWWMERCH_RESULT1, WWWMERCH_RESULT2 or WWWMERCH_RESULT999 structure. By specifying MERCH_RESULT1, MERCH_RESULT2 or MERCH_RESULT999 in the Output Data Format parameter, you inform the API that you are expecting output data in the WWWMERCH_RESULT1, WWWMERCH_RESULT2 or WWWMERCH_RESULT999 style structure, respectively. The data expected from the host service and the transmission method (section 3.8 on page 55) desired will help determine which of these formats to choose for each transaction processed.

The WWWMERCH_RESULT1 structure has the following field elements (Note - The following field elements are formatted using HTML table definitions that are best viewed with a browser that supports tables):

Field Name	DataType	Length	Description
Bytes Provided	binary	9,0	The numeric value of this field represents the amount of space that you have allocated and made available for the returned information from the Merchant/400 API. It must be at least the size of the WWWMERCH_RESULT1 structure and should include additional space for the Return Comment.
Bytes Available	binary	9,0	This value will be returned by the Merchant/400 API and contains the size of the actual data returned. If the Bytes Provided field contains a value that is larger, or the same as, the size of the data returned, Bytes Available will reflect the size of the actual data returned. If the Bytes Provided field contains a value that is smaller than the size of the data that could be returned, Bytes Available will reflect the size of the data that could have been returned and the return data will be truncated to the size of the value in Bytes Provided.
Result Code	character	256	A string that identifies the results of the transaction. This

			code is originated from the credit card processor and should be checked and recorded as the approval identification for the transaction. In the case of a "book" transaction, this code should be saved, as it will be required to complete the charge as part of a future "ship" transaction.
Return Comment Length	binary	9,0	This field contains the size of the comments returned in the Return Comment area. If the Bytes Provided field contains a value that is large enough to accommodate a Return Comment, this field will reflect the actual size of the comment returned, otherwise no comment will be returned and this field will be set to a value of zero (0).
Return Comment	character	????	The Return Comment field is not really part of the WWWMERCH_RESULT1 structure, but an area that is made available for any comments returned by the credit card processor. This area will contain information that is important to why a transaction was denied. If the transaction was successful, this area will contain the string "Success".

The WWWMERCH_RESULT2 structure has the following field elements (Note - The following field elements are formatted using HTML table definitions that are best viewed with a browser that supports tables):

Field Name	DataType	Length	Description
Bytes Provided	binary	9,0	The numeric value of this field represents the amount of space that you have allocated and made available for the returned information from the Merchant/400 API. It must be at least the size of the WWWMERCH_RESULT2 structure and should include additional space for the Return Comment.
Bytes Available	binary	9,0	This value will be returned by the Merchant/400 API and contains the size of the actual data returned. If the Bytes Provided field contains a value that is larger, or the same as, the size of the data returned, Bytes Available will reflect the size of the actual data returned. If the Bytes Provided field contains a value that is smaller than the size of the data that could be returned, Bytes Available will reflect the size of the data that could have been returned and the return data will be truncated to the size of the value in Bytes Provided.
Result Code	character	256	A string that identifies the results of the transaction. This code is originated from the credit card processor and should be checked and recorded as the approval identification for the transaction. In the case of a "book" transaction, this code should be saved, as it will be required to complete the charge as part of a future "ship" transaction.
Result Host Reference	character	256	A string that uniquely identifies the transaction. This code is originated from the DataCash service and should be recorded for future reference and identification for the transaction. In the case of a "book" transaction, this code must be saved, as it will be required to complete the

			charge as part of a future "ship" transaction.
Return Comment Length	binary	9,0	This field contains the size of the comments returned in the Return Comment area. If the Bytes Provided field contains a value that is large enough to accommodate a Return Comment, this field will reflect the actual size of the comment returned, otherwise no comment will be returned and this field will be set to a value of zero (0).
Return Comment	character	????	The Return Comment field is not really part of the WWWMERCH_RESULT2 structure, but an area that is made available for any comments returned by the credit card processor. This area will contain information that is important to why a transaction was denied. If the transaction was successful, this area will contain the string "Success".

The WWWMERCH_RESULT999 structure has the following field elements (Note - The following field elements are formatted using HTML table definitions that are best viewed with a browser that supports tables):

Field Name	DataType	Length	Description
Bytes Provided	binary	9,0	The numeric value of this field represents the amount of space that you have allocated and made available for the returned information from the Merchant/400 API. It must be at least the size of the WWWMERCH_RESULT999 structure and should include additional space for the Return Buffer.
Bytes Available	binary	9,0	This value will be returned by the Merchant/400 API and contains the size of the actual data returned. If the Bytes Provided field contains a value that is larger, or the same as, the size of the data returned, Bytes Available will reflect the size of the actual data returned. If the Bytes Provided field contains a value that is smaller than the size of the data that could be returned, Bytes Available will reflect the size of the data that could have been returned and the return data will be truncated to the size of the value in Bytes Provided.
Return Buffer Length	binary	9,0	This field contains the size of the data returned in the Return Buffer area. If the Bytes Provided field contains a value that is large enough to accommodate a Return Buffer, this field will reflect the actual size of the buffer data returned, otherwise no data will be returned and this field will be set to a value of zero (0).
Return Buffer	character	????	The Return Buffer field is not really part of the WWWMERCH_RESULT999 structure, but an area that is made available for any data that is to be returned by the upline host processor. This area will contain information that is important to the results of a transaction. The data received in this buffer will have been converted appropriately (based on the value of the Results Body CCSID in the WWWMERCH_INPUT999 structure) and will be in the job's CCSID when returned to your application.

3.6 Error Conditions

3.6.1 What to Expect in the Completion Indicator (plResultSuccess) Parameter

As the WwwMrcSubmit/WWWMRCSB API executes, conditions may arise that prohibit it from completing the transaction. These conditions will be monitored and returned in the plResultSuccess parameter. If the results are successful, the value of this parameter will be zero (0). If, for some reason, the transaction could not be completed successfully, a different numeric value will be returned. The table below contains the list of codes that may be returned with the descriptive meaning for each code.

3.6.2 plResultSuccess Values.

ILE Define	Code	Description
WWW_MAPIERR_SUCCESS	0	This value indicates a transaction that has resulted in the successful completion of the charge to the customer's credit card. You should record any information returned that is relative to the card processor's return code or host reference and may continue with the completion of the order.
WWW_MAPIERR_LENINV	1	A length parameter or input / output format field was found to have a value that was too small for use when passed for use by the API. This usually is related to the size of the output format structure. The transaction was not submitted. Check the entry to make sure that it contains a value sufficient for the transaction being processed before attempting the request again.
WWW_MAPIERR_NO_RESULTS	2	No results, or invalid results, were returned from the processing host for the transaction that prevents the continuing execution of the API. The results of the transaction are unknown. Manual transaction checking should be performed to insure that the charge has been approved before the order is completed.
WWW_MAPIERR_INPUT_FORMAT	4	The Input Data Format parameter was invalid when passed for use by the API. The transaction was not submitted. Check the entry to make sure that it matches the list of input formats (Input Formats and Structures. on page 40) and attempt the request again.
WWW_MAPIERR_OUTPUT_FORMAT	5	The Output Data Format parameter was invalid when passed for use by the API. The transaction was not submitted. Check the entry to make sure that it matches the list of output formats (Output Formats and Structures. on page 46) and attempt the request again.
WWW_MAPIERR_TRANSMIT_METHOD	6	The Transmission Method parameter was invalid when passed for use by the API. The transaction was not submitted. Check the entry to make sure that it matches the list of

		transmission methods (section 3.8 on page 55) and attempt the request again.
WWW_MAPIERR_HOST_NAME	31	The host name or IP address specified in the pachRemoteHost parameter could not be found. The transaction was not submitted. It is possible that you are attempting to connect to a host that does not exist. Network or domain name activity may also be a factor in this error. Check the value of the pachRemoteHost parameter before submitting the request again. You may also wish to check your TCP/IP configuration and domain name server configuration (if one exists) to make sure that the host desired has been set up properly.
WWW_MAPIERR_CONNECT_FAILED	32	A connection between Merchant/400 and the remote host system could not be established. The transaction was not submitted. It is possible that you are attempting to connect to a server that does not exist or that is not currently active. Network activity may also be a factor in this error. Check the value of the pachRemoteHost and lRemotePort parameters before submitting the request again. You may also wish to utilize other TCP/IP utilities (ping, etc.) to check your connection capabilities to the host desired.
WWW_MAPIERR_HANDSHAKE_FAILED	33	The handshake between Merchant/400 and the remote host system has failed. The transaction was not submitted to avoid the possibility of sending information in an unprotected manner. It is possible that you are attempting to connect to a server or port that does not support SSL. Check the value of the pachRemoteHost and lRemotePort parameters before submitting the request again.
WWW_MAPIERR_TRANS_REJECTED	51	The transaction has been rejected by the credit card processor or the partner processing agency. You may wish to look at other results found in the output format structure to determine the source of the error before submitting the request again.
WWW_MAPIERR_TRANS_TEMPREJECT	52	The transaction has been temporarily rejected by the credit card processor. You may wish to look at other results found in the output format structure to determine the source of the error before submitting the request again.
WWW_MAPIERR_ACTIVATION	95	The activation code for the product has either not been set, or has been set incorrectly. The transaction was not submitted. Set the activation key for the API (section 1.2 on page 8) and attempt the request again.
WWW_MAPIERR_INTERNAL	96	An internal error has occurred that prevents the continuing execution of the API. The results of the transaction are unknown. Manual transaction checking should be performed to insure that the

		charge has been approved before the order is completed.
WWW_MAPIERR_PARM_INV	97	One of the parameters or input / output format fields was invalid when passed for use by the API. The transaction was not submitted. Check the entries to make sure that they match the list of required information (section 3.7 on page 53) and attempt the request again.
WWW_MAPIERR_PARM_NULL	98	One of the parameters or input / output format fields were not passed or defined for use by the API. The transaction was not submitted. Check the entries to make sure that they match the list of required information (section 3.7 on page 53) and attempt the request again.
WWW_MAPIERR_MALLOC	99	A memory error has occurred that prevents the continuing execution of the API. The results of the transaction are unknown. Manual transaction checking should be performed to insure that the charge has been approved before the order is completed.

3.7 Required Information

3.7.1 Minimum Data to Send to the API

Quite a bit of information has been collected from the customer by your application to process the credit card transaction. Depending on the input and output formats (section 3.5 on page 40) and the transmission method used, it is possible that all the information will not be required.

3.7.1.1 Merchant/400 API Parameters.

Each of the parameters defined for the WwwMrcSubmit API call is required in one form or another. The Remote Host Name, Remote Host Name Length and Remote Port Number parameters are critical for the API to know where to send the request. The Transmission Method, Input Data Format and Output Data Format parameters tell the API how to function, both from the standpoint of the transaction stream and in how it interfaces with the data you have supplied. The other parameters contain information for the API to process or will be returned with results data from the card processor and the API.

It is important to keep in mind that, while some parameters are considered output (returning data from the API) to the calling application, it is still a minimum requirement that they be defined as a placeholder where the API will insert the return data.

3.7.1.2 Fields in the pSubmitInputs Input Structure.

Customer, order, and credit card information is passed to the Merchant/400 API via the pSubmitInputs parameter. Depending on the transmission method (section 3.8 on page 55) to be used, some of the fields are required and some are not.

WWWMERCH_INPUT1 structure fields required for MERCH_METHOD1:

- Host Service Name
- Merchant Name
- Merchant Password
- Order ID
- Customer Name
- Customer Address
- Customer City
- Customer State/Province
- Customer Zip/Postal Code
- Customer Country
- Credit Card Number
- Card Expiration Month
- Card Expiration Year
- Order / Charge Amount

WWWMERCH_INPUT1 structure fields required for MERCH_METHOD2:

- Transaction Type
- Merchant Name
- Merchant Password
- Order ID
- Customer Address

- Customer Zip/Postal Code
- Credit Card Number
- Card Expiration Month
- Card Expiration Year
- Order / Charge Amount
- Approval Code (if this is a "ship" transaction)

WWWMERCH_INPUT2 structure fields required for MERCH_METHOD3:

- Transaction Type
- Merchant Name
- Merchant Password
- Order ID (Host Reference if this is a "ship" transaction)
- Credit Card Number
- Card Expiration Month
- Card Expiration Year
- Card Start/Issue Month (if Card Start/Issue Year is specified)
- Order / Charge Amount
- Approval Code (if this is a "ship" transaction)
- Currency of Charge Amount

3.7.1.3 ***Fields in the pSubmitResults Output Structure.***

The results of the transaction (success or failure) are received from the Merchant/400 API via the pSubmitResults parameter. Here again, the transmission method could play a part in the fields that are required. Currently, the fields required in the WWWMERCH_RESULT1 structure are identical when using either transmission method.

WWWMERCH_RESULT1 structure fields required:

- Bytes Provided

The Bytes Provided field should contain a value large enough to accommodate the other fields in the structure, including space for the Return Comment field. Return Comment should be large enough to receive any comments that might be anticipated coming from the API or the card processor. This is where you will receive information about why the transaction failed and may be integral in determining a course of action for your application (retry the transaction later) or for your customer (call their bank or credit card agency). The actual text is usually determined by the card processor or partner agency.

3.8 What Happens - Transmission Methods

3.8.1 The lTransmitMethod / Transmission Method Parameter is the Key

You, the merchant, collect order information from the customer. This could be done over the Internet with a CGI form-processing program, via an interactive application run by your customer over the Internet with Webulator/400, or by an application run directly on the AS/400 via a "green screen" terminal or a batch job. The following information is collected.

a. Name and address information of the purchaser
b. Credit card number and expiration date
c. Credit card start date or issue number
d. Amount of sale
The value of lTransmitMethod that you use will depend on the route you wish the transaction to take.

3.8.1.1 MERCH_METHOD1 - Routing Transactions Through Holonet IAT.

Many transactions can be routed through Holonet IAT on their way to Credit Card Network or Anacom Communications. Since there is no direct SSL link to Credit Card Network, this is the method to use when communicating with their service. Transactions routed directly to Holonet will also use this method.

1. The order information is packaged with your Merchant name and password (section 2.8 on page 19) by your application into the appropriate structure (Input Formats and Structures. on page 40) and passed to the Merchant/400 API.

2. The Merchant/400 API formats the authorization requests and connects to the SSL transaction provider, Holonet IAT (section 2.4 on page 15). The request is then encrypted and transmitted to Holonet IAT.

3. Based on the Service Name and Merchant name that were passed to the API in the input structure (Input Formats and Structures. on page 40) and transmitted with the request, Holonet IAT forwards the request to the appropriate processing partner. The current partner processing choices are:
   a. Credit Card Network (section 2.3 on page 14)
   b. Anacom Communications (section 2.5 on page 16)
   c. Holonet Holopay (section 2.4 on page 15)
4. Since a connection to Anacom Communications will be discussed below (MERCH_METHOD2 - Routing Transactions to Anacom Communications. on page 56) (and is fairly similar to the methods Holonet IAT uses to communicate with them, we will continue this example using Credit Card Network. At this point in the process, Holonet IAT unencrypts the request and re-packages it into the PGP (Pretty Good Privacy (http://www.pgp.com)) mail encryption format. Once this is done, the request is transmitted via e-mail to Credit Card Network. Card processor information and currency denomination are also determined by the Merchant name.

5. Credit Card Network unencrypts the request and submits the information to the card processor for authorization. This is done with the ICVerify electronic payment program.

6. A reply is returned from the card processor in seconds and the entire process is reversed. Credit Card Network packages the response, using PGP, and sends it to Holonet IAT.

7. When received by Holonet IAT, the response is unencrypted and re-encrypted (into SSL) and sent on to the waiting Merchant/400 API.

8. The Merchant/400 API unencrypts the response and determines if the transaction was accepted or denied. Result information (Output Formats and Structures. on page 46) is formatted in the appropriate structure and returned to your application for further processing and storage.

9. Complete transaction reports including credit card numbers and expiration dates are provided by Credit Card Network in a password protected, secure directory. These reports are produced daily and are html files. They are available to review and download and contain each transaction and daily totals.

10. Credit Card Network performs an automatic settlement each evening whereby all the transactions for the day are settled and funds are transferred by the processor into the merchant's checking account within 48 hours.

Credit Card Network also provides a manual data entry screen in your Merchant password protected directory. This data entry screen can be used to enter transactions that are outside the normal order process.

*3.8.1.2 **MERCH_METHOD2 - Routing Transactions to Anacom Communications.***

Anacom Communications provides a direct SSL link. As a result, communications to Anacom may be routed through this routing method. Another option for communicating with Anacom is MERCH_METHOD1.

1. The order information is packaged with your Merchant name and password (section 2.8 on page 19) by your application into the appropriate structure (Input Formats and Structures. on page 40) and passed to the Merchant/400 API. Transaction type (Transaction Type on page 40) information is also provided.

2. The Merchant/400 API formats the authorization requests, which are encrypted and transmitted to Anacom Communications.

3. Anacom unencrypts the request and submits the information to the card processor for authorization. This is done with the ICVerify electronic payment program.

4. A reply is returned from the card processor in seconds and the process is reversed. Anacom packages the response, using SSL, and sends it to the waiting Merchant/400 API.

5. The Merchant/400 API unencrypts the response and determines if the transaction was accepted or denied. Result information (Output Formats and Structures. on page 46) is formatted in the appropriate structure and returned to your application for further processing and storage.

6. Complete transaction reports are provided in a date search format by Anacom Communications in a password protected, secure directory. They are available to review and download and contain each transaction and transaction count totals.

Anacom Communications also provides a manual data entry screen in your Merchant password protected directory. This data entry screen can be used to enter credit transactions.

*3.8.1.3 **MERCH_METHOD3 - Routing Transactions to DataCash Ltd.***

DataCash Ltd. provides a direct SSL link. As a result, communications to DataCash may be routed through this routing method.

1. The order information is packaged with your Merchant name and password (section 2.8 on page 19) by your application into the appropriate structure (Input Formats and Structures. on page 40)

and passed to the Merchant/400 API. Transaction type (Transaction Type on page 40) information is also provided.

2. The Merchant/400 API formats the authorization requests, which are encrypted and transmitted to DataCash.

3. DataCash unencrypts the request and checks the data sent for validity. This is done with the DataCash Transaction.raw proprietary electronic payment clearing system interface. Initially, a Luhn check is done on the card, and a check that they support that particular card and currency combination. Then they check the card details for validity - whether the expiration date is reasonable, whether the start date and/or issue number look reasonable and are what are expected. The transaction information is then submitted via a dedicated X.25 link to the card processor or acquiring bank for authorization.

4. A reply is returned from the card processor in seconds and the process is reversed. DataCash packages the response, using SSL, and sends it to the waiting Merchant/400 API.

5. The Merchant/400 API unencrypts the response and determines if the transaction was accepted or denied. Result information (Output Formats and Structures. on page 46) is formatted in the appropriate structure and returned to your application for further processing and storage.

6. Complete transaction reports are provided in a date search format by DataCash in a password protected, secure directory. They are available to review and download and contain each transaction and transaction count totals.

DataCash also provides an online service that provides a variety of functions, including a private, manual, data entry screen which can be used to enter credit transactions.

3.8.1.4 MERCH_METHOD999 - Routing Free-Format Transactions.

Free format transmissions are possible using this method with the WWWMERCH_INPUT999 and WWWMERCH_RESULT999 input and output structures. This allows you to format the *HTML form* request within your application and have the Merchnat/400 API work as a transmt/receive gateway to your upline host. As noted in the following steps, very little is done to affect the data streams that are transmitted or received.

1. The form information (following standard HTTP GET, POST, etc. formats) is packaged by your application into the WWWMERCH_INPUT999 structure (Input Formats and Structures. on page 40) and passed to the Merchant/400 API.

2. The Merchant/400 API searches for the characters that separate the HTTP <HEAD> information from the <BODY> information. The <HEAD> information is converted to CCSID 819 (the standard for HTTP <HEAD> information over the Internet) and the <BODY> information is converted to ASCII according to the value of the Input Body CCSID (Input Body CCSID on page 46). Once this is complete, the buffer is transmitted to the upline host.

3. The upline host unencrypts the request and processes the information transmitted.

4. A reply is packaged, using SSL, and returned from the upline host to the waiting Merchant/400 API.

5. The Merchant/400 API unencrypts the response, then finds the <HEAD> and <BODY> information. The <HEAD> information is converted from CCSID 819 and the <BODY> information is converted from ASCII according to the value of theResults Body CCSID (Results Body CCSID on page 46).

Result information is formatted in the WWWMERCH_RESULT999 structure (Output Formats and Structures. on page 46) and returned to your application for further processing and storage.

It is important to note that, while the MERCH_METHOD999 free format method is very flexible, information passed in the WWWMERCH_INPUT999 and WWWMERCH_RESULT999 structures must conform to the HTTP form structure standards. More information about HTTP may be obtained at the World Wide Web Consortium (http://www.w3.org/). The topics that discuss Script Inputs and Script Outputs may also be of assistance.

For questions or comments about Credit Card Network, write to ccnetwork@creditnet.com (mailto:ccnetwork@creditnet.com)

For questions or comments about Holonet IAT, write to support@holonet.net (mailto:support@holonet.net)

For questions or comments about Anacom Communicatioins, write to support@anacom.com (mailto:support@anacom.com)

For questions or comments about DataCash, write to info@datacash.com (mailto:info@datacash.com)

4. The I/NET Store - A Sample Application

4.1 The I/NET Store - A Sample Application

4.1.1 Introduction

The I/NET Store is provided with Merchant/400 to provide the following:

- A working application that can be used to set up a mail order shop on the web
- A demonstration of how the API could be used in a custom application

The store is an application that allows the user to select one or more products to order from a list of items that you have for sale. It then calculates the total cost for the customer, allowing the customer to accept the order or to change the items that are to be purchased. When the customer is satisfied with the list of items to be ordered, they enter credit card, personal and address information and then submit the completed order. At this time the credit card is validated using the Merchant/400 API. If the credit card is accepted, the order is stored and the user is given a screen that thanks them for their order. If the validation fails, the user is prompted to correct the information so the validation can be attempted again.

The list of items that the customer selects for purchase is stored using a cookie. A cookie is a piece of information that is stored by the browser. This allows the customer to leave the store and return later with the items that were selected earlier still in their "shopping basket". The list of items remains until a customer completes an order, or closes the browser. For more information on cookies see Cookies (section 4.7 on page 79).

The application only charges amounts to credit cards via "sale" transactions. That is, it charges the credit card immediately upon verification. For more information on transaction types see Transaction Type (Transaction Type on page 40).

The application can be configured to fit your requirements. For information on configuring the store see Configuration (section 4.2 on page 62).

4.1.2 Store Screens

The store consists of four screens that the user interacts with. These screens can be customized to look the way you want them to. All of the text displayed is stored in text files (section 4.4 on page 71) and message files (section 4.5 on page 73) to allow it to be changed to fit your needs.

4.1.2.1 *Order Form*

The first screen that the user sees is the Order Form. This screen presents the customer with a list of products that can be ordered. This list displays the product identifier, description, price, unit of measure, and an optional picture for each product. The user types the number of each item to be ordered. When the user has selected everything to be ordered, he/she clicks the "Cash Register" button to complete the order.

This list of items that are displayed is stored in the Product Description physical file (MPRODCTP) (Product Description Physical File on page 66).

4.1.2.2 ***Cash Register***

After the customer has selected all of the items to be purchased, he/she is shown this screen. Listed on the screen is the list of all the items that were selected to be ordered and a total, including shipping and handling. If the customer wants to finish the order, he/she clicks the "Yes" button, otherwise the "No" button is clicked, which causes the Order Form to be redisplayed, allowing the customer to change their selections.

The amounts to be charged for shipping and handling is stored in the global configuration physical file (MGLOBALP) (GFRTI on page 65).

4.1.2.3 ***Check Out***

When the customer has confirmed that the items to be ordered are correct, the Check Out screen is displayed. This screen prompts for several types of information:

- Credit card information, including card type, number and expiration date.
- Customer information, including name, company, and e-mail address
- Address information, including the billing address and shipping address. the shipping address is only needed if it is different than the billing address.

Note: The customer name and billing address must match the information from the credit card.

After the customer fills in the information, he/she presses "Complete Order". At this time, the store attempts to validate the credit card using the Merchant/400 API (section 4.6 on page 77). If the card is accepted, the order information is stored in the customer order files (MORDERP and MORDITMP) (Customer Order Physical File on page 67). If the card is not accepted, the Check Out screen is redisplayed with an error from the credit card processor to allow the customer to correct the problem and try again.

4.1.2.4 ***Thank You***

When the order is complete. A screen is displayed that shows the order number and the approval code from the credit card processor.

4.2 Configuration

4.2.1 Introduction

The sample shopping application is configured, when installed, to run from the WWWMRCH library and to locate all the IFS stream files in /WWWSERV/WEBDOCS/SHIPPED/MERCHANT. The application can be run for testing from these locations, but if another version of Merchant/400 is installed, these will be overwritten. If you plan to use the sample application for a store of your own. please follow the complete directions below.

The sample store is configured when installed to call a test processing center at I/NET. This server will do minimal validation of the data passed to it and will either return a success or failure, allowing the API to be tested without incurring transaction fees. The test processing center only accepts one test credit card. The credit card is: Master Card, 5499740000000032, and expires 1/1999. This is not a real credit card.

4.2.2 Running the Sample Store (for Testing Only)

To run the sample from the installed locations, you will need to the following:

1. Add the WWWMRCH library to the Commerce Server/400 library list. To do this you will need to add WWWMRCH to your library list using the ADDLIBLE command before starting Commerce Server/400. You will to have Commerce Server/400 configured to use the current library list as its library list for this to work. This can be set by calling CHGWWWCFG INLLIBL(*CURRENT).
2. Add the WWWMRCH library to the Commerce Server/400 script libraries. Run the ADDWWWSCPL command to do this. See the Script Libraries documentation for more information.
3. Add the ability to serve both HTTP and SSL documents from the WWWMRCH library. Run the CHGWWWCFG and WRKWWWDIR command to do this. See the setup documentation for more information about enabling Commerce Server/400 to do SSL.

You should now be able to run the sample store. Call the following URL to open the store from your browser (substituting your servers name for www.server.com):

http://www.server.com/qsys/wwwmrch/merchant.

You should see the Order Form. Select one or more products to buy and click "Cash Register". On the Cash Register screen, click "Yes". You should now see a secured Product Check Out entry form. Select "Master Card" number 5499740000000032 with an expiration date of 1/1999. Fill in the rest of the customer information and click "Complete Order". You then should see a Thank You screen. You have just completed a test transaction.

Please do not enter a real credit card number, as this will be transmitted to our test processor at I/NET , and we do not want to know your credit card information.

If you want to test a transaction method of MERCH_METHOD2 (Anacom Communications), you will need to follow the instructions listed above, plus the following:

1. Change the Transmission Method (GTMETH) (GTMETH on page 66) in the global configuration physical file (MGLOBALP) to MERCH_METHOD2.
2. Make sure the Merchant Name (GMRCID) (GMRCID on page 65) in the global configuration physical file (MGLOBALP) is set to "merchantid". Our test processor will only accept requests from this Merchant Name.

4.2.3 Configuring the Sample Store for a Production Environment

To configure the store for a production environment you will need to move all of the files to a new location and to configure the application to find them. You will also need to change the IFS files that contain the HTML that is displayed to represent your company. Last, you will need to update the product database to contain the products or services that you will have for sale in your online store..

The following is a list of steps that need to be performed to configure a store.

1. Copy the WWWMRCH library. This library contains the various programs and databases needed to run the application. You can copy the library using the CPYLIB command. For the remainder of this documentation, it will be assumed that the new library name will be MERCHANT. If it has another name, just use that library in place of MERCHANT.
   Note that the CPYLIB command will create the new objects with you as the owner and WWWUSER with *ALL object authority. This authority will allow the application to run, but you may want to change the owner back to WWWUSER.
2. Add the MERCHANT library to the Commerce Server/400 library list. To do this you will need to add MERCHANT to your library list using the ADDLIBLE command before starting Commerce Server/400. You will to have Commerce Server/400 configured to use the current library list as its library list for this to work. This can be set by calling CHGWWWCFG INLLIBL(*CURRENT). If you added the WWWMRCH library to the library list earlier, you will need to remove it. Note: the database files are found through the library list, which causes a restriction that only one store can be running on a single instance of Commerce Server/400.
3. Add the MERCHANT library to the Commerce Server/400 script libraries. Run the ADDWWWSCPL command to do this. See the Script Libraries documentation for more information.
4. Copy the contents of /WWWSERV/WEBDOCS/SHIPPED/MERCHANT directory to another directory. For the remainder of this documentation, it will be assumed that the new directory will be /WWWSERV/WEBDOCS/MERCHANT. If the document root of your Commerce Server/400 is not /WWWSERV/WEBDOCS then you should copy the file to a directory under your document root. This directory contains the HTML that is displayed at the top and bottom of the application and a variety of GIF files used by the application.
5. Change the IFS File Path (GIFSPT) (GIFSPT on page 66) in the global configuration physical file (MGLOBALP) to /WWWSERV/WEBDOCS/MERCHANT. This can be done with DFU.
6. Edit the new HTML header and footer files. See Editing the Store Headers and Footers (section 4.4 on page 71) for more information.
7. Edit the MRCHMSGF message file (section 4.5 on page 73) in the MERCHANT library. The location of the GIF files used in the application are stored in message descriptions. Change the message text for messages MRC0400 through MRC0406 to reflect the new location. MRC0400 should look like the following: ’<IMG SRC="/MERCHANT/MASTERCD.GIF" alt="MasterCard" WIDTH=102 HEIGHT=63 BORDER=0>’, if you copied the GIF files into a merchant directory under your document root.
   Other messages in this message file can be changed in a similar fashion to customize your store.
8. Update the global configuration physical file (MGLOBALP) (Global Configuration Physical File on page 65). This physical file contains a single record that stores a variety of information that the application needs to run. This also where you set many of the Merchant/400 API parameters, like host service name, transmission method, partner host name, and partner SSL port, that will be used for your credit card transactions..
9. Add the products or services that you will include in your store to the Product Description physical file (MPRODCTP) (Product Description Physical File on page 66)

4.3 The Store Database

4.3.1 Global Configuration Physical File

The MGLOBALP physical file contains a single record that defines how the store operates. This file must contain a record, and if more than one exists, the first is used.

This file contains your Merchant Name and Password, so be careful with the authority given to this file. WWWUSER must have read access to this file.

Field Name	Data Type	Length	Description
GFRTI	zoned decimal	7,2	This is the freight charge for each item in the customers order. This can be zero, in which case no per item charge is added.
GFRTF	zoned decimal	7,2	This is the flat freight charge added to each order. This can be zero, in which no per order charge is added.
GMRCID	character	256	This is the merchant name that is passed to the Merchant/400 API. See the Input and Output Formats (Merchant Name on page 44) for more information on setting this value.
GMRCPW	character	256	This is the merchant password that is passed to the Merchant/400 API. See the Input and Output Formats (Merchant Password on page 44) for more information on setting this value.
GSVCNM	zoned decimal	3,0	This is the remote processing host service name. See the Input and Output Formats (Host Service Name on page 42) for more information on setting this value.
GHOSTN	character	256	This is the name and domain, or IP address, of the remote processing host. See the Using the Merchant/400 API in an ILE Program (section 3.3 on page 35) or an OPM Program (`Remote Host Name` on page 38) for more information on setting this value.
GPORT	zoned decimal	5,0	This is the socket port number used by the remote host for SSL communications. This will usually be set to 443. See the Using the Merchant/400 API in an ILE Program (`lRemotePort` on page 35) or an OPM Program (`Remote Port Number` on page 38) for more information on setting this value.
GHPORT	zoned decimal	5,0	This is the socket port used by Commerce Server/400 for non-secure communication (HTTP). This is used by the store application to switch to/from secure mode when needed. This will usually be set to 80.
GSPORT	zoned decimal	5,0	This is the socket port used by Commerce Server/400 for secure communication (SSL). This is used by the store application to switch to/from secure mode when needed. This will usually be set to 443.

GIFSPT	character	256	This is the IFS file path to where the files containing the header and footer text files for the application. This is set to '/WWWSERV/WEBDOCS/SHIPPED/MERCHANT' when the product is installed and should be changed to reflect the location where your unique text files are stored.
GTMETH	zoned decimal	3,0	This tells the Merchant/400 API how to the transmission method to use when transmitting the data. See the Using the Merchant/400 API in an ILE Program (`lTransmitMethod` on page 35) or an OPM Program (`Transmission Method` on page 38) for more information on setting this value.
GVISAY	character	1	This indicates that the merchant can accept VISA credit cards. Set to 'Y', 'y', or '1' for YES. Any other values indicates NO.
GMCY	character	1	This indicates that the merchant can accept Master Card credit cards. Set to 'Y', 'y', or '1' for YES. Any other values indicates NO.
GDISCY	character	1	This indicates that the merchant can accept Discover credit cards. Set to 'Y', 'y', or '1' for YES. Any other values indicates NO.
GAMEXY	character	1	This indicates that the merchant can accept American Express credit cards. Set to 'Y', 'y', or '1' for YES. Any other values indicates NO.
GTRTYP	zoned decimal	3,0	This tells the Merchant/400 API the transaction type to use for the credit card transaction. This must be set to MERCH_TRTYPE1_SALE or MERCH_TRTYPE1_BOOK. See the Input and Output Formats (Transaction Type on page 40) for more information on setting this value.

4.3.2 **Product Description Physical File**

The MPRODCTP physical file contains information describing each product that is being sold in the store.

There are two logical files over this file:

- MPRODCTL1, sorted by Product ID (PRDID)
- MPRODCTL2, sorted by Sequence Number (PRDSQ) and Product ID (PRDID)

Field Name	Data Type	Length	Description
PRDID	character	10	This is the identifier assigned to the product. It can be any value up to 10 characters long. The product identifier cannot contain any of the following characters: ',', '&', or ';'.
PRDSQ	zoned decimal	6,0	This field determines the order that products are displayed in the store. The products are displayed in

			ascending order. These values do not have to be sequential.
PRDNM	character	25	This is then name of the product.
PRDPRC	zoned decimal	7,2	This is the price for each unit of the product. If more than one of the product is ordered, this value is charged for each product ordered.
PRDWT	zoned decimal	3,0	This is the weight for each unit of the product.
PRDUM	character	10	This is a description for each units packaging. Examples include: each, set, pair, case, pounds, liters, etc.
PRDPIC	character	256	This is the HTML displayed in the first column for the product. It can contain an image tag and/or other text to display. An example: <img src="/products/product.gif" alt="Product Name"> Note that any file paths in the HTML must be relative to the document root (i.e. begin with a slash).
PRDURL	character	256	This is the URL of an HTML document that gives more information about the product. If this value is supplied, the Product ID becomes a link to the description HTML document. If no URL is supplied then the Product ID is displayed without a link. An example: /products/product.html Note that this URL must be relative to the document root (i.e. begin with a slash).
PRDDS1	character	30	The five product description lines are combined into a single description and displayed under the product name. Each description line is separated by a space from the preceding one, allowing them to wrap into a single block of text.
PRDDS2	character	30	See PRDDS1.
PRDDS3	character	30	See PRDDS1.
PRDDS4	character	30	See PRDDS1.
PRDDS5	character	30	See PRDDS1.

4.3.3 Customer Order Physical File

The MORDERP physical file contains information about the orders placed by customers. It contains one record for each order that has been successfully placed. It contains customer information, totals calculated for the sale, and several fields that can be used to track the order after placement. The shipping address is only supplied if it is different than the billing address.

This file contains your customers credit card information, so be careful with the authority given to this file. WWWUSER must have change access to this file.

There is one logical file over this file:

- MORDERL1, sorted by Order ID (ORDID)

Field Name	Data Type	Length	Description
ORDID	character	10	This is the order identifier that is assigned to the order.

			These are assigned sequentially and are assigned before the credit card is verified. No order record is stored if the credit card verification fails, so this file can and will contain gaps in the sequence of IDs. The last ID used is stored in the MORDIDP table. This value is read, incremented, and then written back to the file. If no record exists in the MORDIDP table, an order ID of 1 is used and written to the file.
ORDDT	zoned decimal	8,0	This is the date that the order was placed. It is stored in the format of YYYYMMDD.
ORDCNM	character	40	This is the name of the customer.
ORDCCO	character	40	This is the name of the company.
ORDCA1	character	40	This is the first line of the customers billing address.
ORDCA2	character	40	This is the first line of the customers billing address, if needed.
ORDCCT	character	40	This is the billing address city.
ORDCST	character	2	This is the billing address state.
ORDCZP	character	9	This is the billing address ZIP code.
ORDCCY	character	20	This is the billing address country.
ORDCPH	character	20	This is the customer's phone number.
ORDCFX	character	20	This is the customer's FAX number.
ORDCEM	character	40	This is the customer's e-mail address.
ORDCRF	character	40	This is the reference number that the customer entered. It is only for the customers use to allow them to identify the order.

ORDCDT	character	2	This is the credit card type.
			
			`MC - Master Card`
			`V - VISA`
			`D - Discover`
			`AM - American Express`

ORDCDN	character	19	This is the credit card number
ORDCDX	zoned decimal	8,0	This is the credit card expiration date. It is stored in the format of YYYYMMDD, where February 1999 is stored as 19990201.
ORDSA1	character	40	This is the first line of the customers shipping address.
ORDSA2	character	40	This is the first line of the customers shipping address, if needed.
ORDSCT	character	40	This is the shipping address city.
ORDSST	character	2	This is the shipping address state.
ORDSZP	character	9	This is the shipping address ZIP code.
ORDSCY	character	20	This is the shipping address country.
ORDSDT	zoned decimal	8,0	This field can be used to store the date that the order was shipped. It is left blank by the store application.
ORDSCA	character	40	This field can be used to store what carrier shipped the order. This field is left blank by the store application.
ORDSBL	character	40	This field can be used to store the bill of laiding number. This field is left blank by the store application.
ORDAUT	character	20	The authorization ID returned by the Merchant/400 API. See the Input and Output Formats (Result Code on page 47) for more information on this value.

ORDFRT	zoned decimal	7,2	This is the total freight costs charged for the order. It is equal to the flat freight charge plus the per item freight charge time the number of items ordered.
ORDTOT	zoned decimal	9,2	This is the total amount charged for the order. It is the total charged for the item(s) ordered, plus the freight charges.

4.3.4 Customer Order Item Physical File

The MORDITMP physical file contains one record for each type of product that a customer ordered. If the customer orders one of product A and two of product B, two records will be created, a record for product A with a quantity of one and a record for product B with a quantity of two.

Field Name	Data Type	Length	Description
ORDID	character	10	This is the order identifier.
PRDID	character	10	This is the product identifier.
ORDQTY	zoned decimal	6,0	This is the number of items of this product that were ordered.
PRDPRC	zoned decimal	7,2	This is the price of the item that was ordered. This is the price for one item of this type, so the total cost for all of the items of this type is equal to this quantity multiplied the number of items ordered. This is the cost of the item at the time that it was ordered.

4.3.5 The Order Identifier Control Physical File

The MORDIDP physical file contains zero or one records that control the generation of order numbers. If the file contains zero records, the next order number to be assigned will be 1. If a record does exist, it contains the last number generated, and the store application reads this value, increments it and writes back the new number.

Field Name	Data Type	Length	Description
ORDID	character	10	This is the last order identifier that was assigned to an order.

4.3.6 Net.Commerce Order Information Physical File

The MNCINFP physical file contains information about the orders placed by customers via the Net.Commerce product from IBM. It is used by the Net.Commerce - Merchant/400 do_payment interface (section 2.10 on page 21) to help communicate information about completed credit card transactions to a Net.Commerce shopper. It contains one record for each order that has been successfully placed and processed by Merchant/400. It contains fields that can be used to track the order after placement.

This file contains your customer's credit card information, so be careful with the authority given to this file.

There is one logical file over this file:

- MNCINFL1, sorted by Order ID (ORDID)

Field Name	Data Type	Length	Description
ORDID	character	10	This is the order identifier that is assigned to the order.

			These are assigned by Net.Commerce before the credit card is verified and can be associated with the order_rn Net.Commerce variable. No MNCINFP record is stored if the credit card verification fails, so this file may contain gaps in the sequence of IDs.
ORDCDN	character	19	This is the credit card number
ORDCDX	zoned decimal	8,0	This is the credit card expiration date. It is stored in the format of YYYYMMDD, where February 1999 is stored as 19990201.
ORDAUT	character	20	The authorization ID returned by the Merchant/400 API. See the Input and Output Formats (Result Code on page 47) for more information on this value.
ORDTOT	zoned decimal	9,2	This is the total amount charged for the order.
ORDREF	character	20	A string that uniquely identifies the transaction. This is the order authorization reference originated from the DataCash servuce and returned by the Merchant/400 API.
ORDCDI	zoned decimal	8,0	This is the credit card start date / issue code. It is stored in the format of YYYYMMDD, where February 1999 is stored as 19990201. In the case of using a card issue number, this will typically be stored in a format zzzznnzz with a value that looks like 0000nn00 (where the nn is the issue code of 01-99).

4.4 Editing the Store Headers and Footers

4.4.1 **Customizing How the Store Looks**

The store application allows you to customize what appears at the top and bottom of each screen of the application. The HTML displayed is stored in IFS text files, and can be modified to customize how the application looks. There are two files for each screen, one containing the top and the other the bottom of the HTML. For instruction on how to edit these files see Creating Content in the Commerce Server/400 documentation.

4.4.1.1 HTML Headers

The HTML headers contain whatever information you want to include at the top of the application screen. You can include a company logo, store information, instructions or other information that will make your store unique.

The headers must include the following tags:

```
<HTML><HEAD>
<TITLE> (The title goes here) </TITLE>
</HEAD>
<BODY (Optional BODY Attributes)>
```

The following sample header sets the title displayed on the title bar, sets the background color to gray, and displays a title in the HTML at the top of the form:

```
<HTML><HEAD>
<TITLE>Order Form</TITLE>
</HEAD>
<BODY BGCOLOR=#C0C0C0>
<CENTER><H1>Order Form</H1></CENTER>
<HR WIDTH="50%">
```

If no header file is found then this default header is used:

```
<HTML><HEAD>
</HEAD>
<BODY>
```

4.4.1.2 HTML Footers

The HTML footers contain the information you want to include at the bottom of the application screen. You can include links to other locations on your site and e-mail addresses, or other information that will help the user use your store efficiently.

The footer files must include the following tags:

```
</BODY></HTML>
```

The following sample footer displays a link to the home page and to the webmaster at the bottom of the form:

```
<P><HR WIDTH="50%"><P><CENTER>
<A HREF="/index.html">Home</A> |
<A HREF="mailto:webmaster@xyz.com">WebMaster</A>
</CENTER>
</BODY></HTML>
```

If no footer file is found then this default header is used:

```
</BODY></HTML>
```

4.4.1.3 ***HTML Files***

The HTML files need to be located in the IFS File Path (GIFSPT) (GIFSPT on page 66) configured in the in the global configuration physical file (MGLOBALP).

This table identifies which HTML text files belong with each screen of the application.

Form Name	Header File	Footer File
Order Form	MRCHOFHD.TXT	MRCHOFFT.TXT
Cash Register	MRCHCRHD.TXT	MRCHCRFT.TXT
Check Out	MRCHCIHD.TXT	MRCHCIFT.TXT
Thank You	MRCHTYHD.TXT	MRCHTYFT.TXT

4.5 The Store Message File

4.5.1 Store Label Text and Messages

All of the text displayed in the application can be customized. This allows you to configure the store to look the way you want it to, and allows you to translate it into a language other than English.

All of the user readable text used in the store application, except the HTML Headers and Footers (section 4.4 on page 71), are stored in a message file (MRCHMSGF) which is found using the Commerce Server/400 library list. All of the message text can be customized for your store. You can add HTML tags to the text if desired, however, it is recommended that only text formatting tags (bold, italics, etc.) be used.

4.5.2 Message Groups

The following is a description of each group of messages.

4.5.2.1 ***MRC0100 - MRC0119***

This is the text that is displayed on the Order Form.

ID	Description
MRC0100	This is the product column heading.
MRC0101	This is the product ID column heading.
MRC0102	This is the description column heading.
MRC0103	This is the price column heading.
MRC0104	This is the unit column heading.
MRC0105	This is the quantity column heading.
MRC0106	This is the cash register button text.
MRC0107	This is the reset button text. It is used for the reset buttons found on all the screens.
MRC0110	These are the table start HTML tags. This can be used to set the width of the table, cell spacing, etc.
MRC0111	This is the product separator. It is displayed between each product. As this is displayed in the table, it will need to have the appropriate <TR> and <TD> tags.
MRC0112	These are the table end HTML tags.
MRC0113	This text is inserted at the front of each heading cell. It can be used to change the formatting of the text. If it contains only spaces it is ignored.
MRC0114	This text is inserted at the end of each heading cell. If it contains only spaces it is ignored.
MRC0115	This text is inserted at the front of each cell. It can be used to change the formatting of the text. If it contains only spaces it is ignored.
MRC0116	This text is inserted at the end of each cell. If it contains only spaces it is ignored.

4.5.2.2 ***MRC0120 - MRC0139***

This is the text that is displayed on the Cash Register.

ID	Description
MRC0120	This is the product ID column heading.

MRC0121	This is the cost column heading.
MRC0122	This is the total label.
MRC0123	This is the shipping and handling label.
MRC0124	This is the total cost label.
MRC0125	This is the prompt that asks the user if he/she has completed shopping.
MRC0126	This is the yes button text.
MRC0127	This is the no button text.
MRC0130	These are the product table's start HTML tags. This can be used to set the width of the table, cell spacing, etc.
MRC0131	These are the product table's end HTML tags.
MRC0132	These are the button table's start HTML tags. This can be used to set the width of the table, cell spacing, etc.
MRC0133	These are the button table's end HTML tags.
MRC0134	This text is inserted at the front of each heading cell. It can be used to change the formatting of the text. If it contains only spaces it is ignored.
MRC0135	This text is inserted at the end of each heading cell. If it contains only spaces it is ignored.
MRC0136	This text is inserted at the front of each cell. It can be used to change the formatting of the text. If it contains only spaces it is ignored.
MRC0137	This text is inserted at the end of each cell. If it contains only spaces it is ignored.

4.5.2.3 ***MRC0140 - MRC0199***

This is the text that is displayed on the Check Out screen.

ID	Description
MRC0140	This is the credit card information heading.
MRC0141	This is the credit card number label.
MRC0142	This is the expiration date label.
MRC0150	This is the customer information subheading.
MRC0151	This is the name label.
MRC0152	This is the company label.
MRC0153	This is the phone number label.
MRC0154	This is the fax number label.
MRC0155	This is the e-mail address label.
MRC0156	This is the reference number label.
MRC0170	This is the billing address subheading.
MRC0171	This is the shipping address subheading.
MRC0172	This text follows the shipping address subheading, indicating that it should be filled in if the shipping address is not the same as the billing address.
MRC0173	This is the complete order button text.
MRC0180	This is the address label.
MRC0181	This is the city label.
MRC0182	This is the state label.
MRC0183	This is the ZIP code label.
MRC0184	This is the country label.
MRC0190	These are the credit card table's start HTML tags. This can be used to set the width of the table, cell spacing, etc.
MRC0191	These are the credit card table's end HTML tags.
MRC0192	These are the customer information table's start HTML tags. This can be used to set the width of the table, cell spacing, etc.

MRC0193	These are the customer information table's end HTML tags.
MRC0194	This text is inserted at the front of each label cell. It can be used to change the formatting of the text. If it contains only spaces it is ignored.
MRC0195	This text is inserted at the end of each label cell. If it contains only spaces it is ignored.
MRC0196	This text is inserted at the front of each cell. It can be used to change the formatting of the text. If it contains only spaces it is ignored.
MRC0197	This text is inserted at the end of each cell. If it contains only spaces it is ignored.

4.5.2.4 ***MRC0200 - MRC0219***

This is the text that is displayed on the Thank You screen.

ID	Description
MRC0200	This message is used to display the order number.
MRC0201	This message is used to display the authorization ID.
MRC0202	This message is used to display the amount charged to the credit card.

4.5.2.5 ***MRC0300 - MRC0399***

The text in this range of messages are general purpose and will be used throughout in the store.

ID	Description
MRC0300	This is the text displayed before an error message.
MRC0301	This is the text displayed before a warning message.
MRC0302	This is the text that shows how a MM/YYYY date should be formatted.
MRC0303	This is the text that shows how a MM/DD/YYYY date should be formatted.
MRC0304	This is the currency symbol that is displayed.
MRC0305	This is the text that shows the minimum number of characters that can be entered.
MRC0306	This is the text that is used to separate a label from it's value.

4.5.2.6 ***MRC0400 - MRC0499***

This is the text that contains the HTML tags for the GIF or JPEG files displayed in the application. You can modify the message text if you wish to display a different image, or replace it with text that does not contain an IMG tag, if you do not wish to display images.

The following is an example of an image tag and a description of its parts:

```
    '<IMG SRC="/MERCHANT/MASTERCD.GIF" ALT="MasterCard" WIDTH=102
HEIGHT=63 BORDER=0>'
```

- SRC - Image to display. Note that this must be relative to your Commerce Server/400 document. It must contain a leading slash.
- ALT - Text to display if the browser does not display images.
- WIDTH - The width in pixels to display the image on the browser. If this is not the same as the width of the image, the image is stretched or shrunk to fit.
- HEIGHT - The height in pixels to display the image on the browser. If this is not the same as the height of the image, the image is stretched or shrunk to fit.

- BORDER - The width of the border around the image in pixels. If you do not want a border, set this to 0.

ID	Description
MRC0400	This is the MasterCard image tag.
MRC0401	This is the VISA image tag.
MRC0402	This is the Discover image tag.
MRC0403	This is the American Express image tag.
MRC0404	This is the error image tag.
MRC0405	This is the warning image tag.
MRC0406	This is the required image tag.

4.5.2.7 ***MRC1000 - MRC1199***

This range contains messages that are displayed in the application. Be careful when modifying these that you keep the same number of replacement fields in the messages. The order of the replacement fields can be changed in the body of the text, meaning that &1 can come after &2. Keep in mind that the actual data field must remain constant, though.

4.6 How the API Fits In

4.6.1 Using the API

Using the API in the store demo application was a simple process. In fact, a total of thirty-one lines of 'C' code was all that was necessary to call the API.

The steps that are done in the application are as follows:

1. Initialize the input structure.
2. Assign the Host Service Name (Input Formats and Structures. on page 40), Merchant Name (Merchant Name on page 44), and Merchant Password (Merchant Password on page 44) to the input structure. These values are read from the global configuration physical file (MGLOBALP) (Global Configuration Physical File on page 65).
3. Assign the Order ID (Order ID on page 44) to the input structure.
4. Assign the Customer Name (Customer Name on page 44), Address (Customer Address on page 44), City (Customer City on page 44), State/Province (Customer State/Province on page 44), Zip/Postal Code (Customer Zip/Postal Code on page 44), and Country (Customer Country on page 44) to the input structure. These values are pulled directly from the Check Out screen.
5. Assign the Order / Charge Amount (Order / Charge Amount on page 45) to the input structure. This is calculated from the items ordered and the shipping and handling charges.
6. Assign Credit Card Number (Credit Card Number on page 44), Card Expiration Month (Card Expiration Month on page 44), Card Expiration Year (Card Expiration Year on page 44) to the input structure. These values are pulled directly from the Check Out screen.
7. Allocate a buffer for the output structure. The buffer was allocated to be large enough to hold the output structure plus 512 additional bytes for the Return Comment.
8. Assign Bytes Provided (Bytes Provided on page 47) in the output structure to the size of the allocated buffer
9. Assign the Bytes Available (Bytes Available on page 47) and Return Comment Length (Return Comment Length on page 48) in the output structure to zero.
10. Call the API. Pass to it the Remote Host, Remote Port, and Transmit Method, along with the input and output structures. The Remote Host, Remote Port, and Transmit Method are all read from the global configuration physical file (MGLOBALP) (Global Configuration Physical File on page 65). See the Using the Merchant/400 API in an ILE Program (section 3.3 on page 35) or an OPM Program (section 3.4 on page 38) for more information on calling the API.
11. Check the return code.
    - If WWW_MAPIERR_SUCCESS is returned, get the Result Code (Result Code on page 47) from the output structure and store it with the order.
    - Otherwise, redisplay the Check Out Screen with an error message that contains the Return Comment (Return Comment on page 48) from the output structure.

4.7 Cookies

4.7.1 **What Is a Cookie?**

A "cookie" is a small piece of information which a CGI script can store with a web browser and later read back from that browser.

To create a cookie, an application sends a "Set-Cookie" HTTP header line like this one in response to a URL access from a browser:

```
    Set-Cookie: NAME=VALUE; expires=DATE; path=PATH; domain=DOMAIN_NAME;
secure
```

- NAME and VALUE are the actual information you're including in the cookie.
- DATE is the time at which the cookie information expires and will be "forgotten" by the browser.
- DOMAIN is a host or domain name for which the cookie is valid.
- PATH specifies a subset of the URLs at that server for which the cookie is valid.
- If you include "secure" in your cookie, then the cookie will only be transmitted over an SSL connection.

All of these fields except NAME=VALUE are optional. It is recommended to set the DOMAIN and PATH, as some browsers do not recognize the cookie, if they are not set.

Whenever the browser sends an HTTP request for a URL on a server for which it has stored cookies, it includes a line of the form:

```
    Cookie: NAME=VALUE; NAME=VALUE; ...
```

which lists all cookies that apply, which can be read by calling the Web Server/400 Get Environment Variable API.

4.7.2 **Examples of Using Cookies**

4.7.2.1 ***Setting a Cookie***

Here is an example of the HTML generated by the store application to set the cookie:

```
    Content-type: text/html
    Set-Cookie: ITEMS=PROD1,1&PROD2,2; path=/qsys/merchant/;
domain=www.xyz.com

    <HTML>...</HTML>
```

This sets a cookie that stores that the user selected one of PROD1 and two of PROD2. Since there is no expiration date, the cookie will disappear when the customer closes the browser.

4.7.2.2 ***Reading a Cookie***

To read the cookie the store calls the Web Server/400 Get Environment Variable API. It passes in an environment variable of "HTTP_COOKIE" which returns the following:

```
    ITEMS=PROD1,1&PROD2,2
```

4.7.2.3 ***Deleting a Cookie***

To delete a cookie, you can set the value of the cookie to blank. Here is an example of the HTML generated by the store application to clear the cookie after the order has been completed:

```
Content-type: text/html
Set-Cookie: ITEMS=; path=/qsys/merchant/; domain=www.xyz.com

<HTML>...</HTML>
```

4.7.3 **Advantages and Disadvantages of Using Cookies**

We used cookies in our application to allow us to store what the user has selected to be ordered. We used cookies for the following reasons:

- The data becomes attached to the customer. If he/she starts to make an order and then jumps to a different HTML document and returns later, the items that had been selected earlier are still selected.
- The Order Form could be on several screens, with each screen adding information to the cookie.

The disadvantages of using cookies for this application are:

- If a user has an old browser that doesn't support cookies, this application will not work.
- In some browsers a user can turn on a warning everytime an application attempts to write a cookie. If the user accepts the cookie, everything works fine.

5. Index

A

Accounts
  Setting Up · 20
Activation Key · 8
Anacom · 11, 15, 16, 20, 40, 42, 43, 55, 56, 58, 62
API
  Calling the · 23, 24, 35, 50, 53
  do_payment · 21, 26, 28
  Input Formats and Structures · 35, 36, 38, 40, 50, 53, 55, 56, 57, 77
  Output Formats and Structures · 36, 39, 46, 47, 50, 54, 56, 57, 58
  Transmission Methods · 35, 36, 38, 39, 53, 55, 56, 62

C

Calling the API · 23, 24, 35, 50, 53
Configuring
  the I/NET Store · 63
Cookies · 60, 78
  Advantages and Disadvantages · 79
Credit Card Network · 11, 14, 15, 20, 40, 42, 55, 56, 58
Credit Card Terminal · 18
Customer Order Item Physical File · 61, 69
Customer Physical File · 61, 67
Customizing the I/NET Store · 71

E

Errors and Return Codes
  do_payment · 21
  Merchant/400 · 50

G

Global Configuration Physical File · 22, 23, 24, 61, 62, 63, 65, 72, 77

H

Holonet · 11, 15, 20, 40, 42, 55, 58

I

I/NET Store · 59, 60
  Configuring · 63
  Customizing · 71
Input Formats and Structures · 35, 36, 38, 40, 50, 55, 56, 57, 77
Installing · 6
  Prerequisites · 6

M

Merchant ID · 19, 20

N

Net.Commerce
  configuring Merchant/400 to use with · 23
  do_payment API · 21, 26, 28
  Net.Data macros · 21, 22, 23, 24, 26, 27, 28, 29

O

Order Identifier Control Physical File · 67, 69
Output Formats and Structures · 36, 39, 46, 50, 56, 57, 58

P

Processing Center's
  Anacom · 11, 15, 16, 20, 40, 42, 43, 55, 56, 58, 62
  Credit Card Network · 11, 14, 15, 20, 40, 42, 55, 56, 58
  Holonet · 11, 15, 20, 40, 42, 55, 58
Product Description Physical File · 60, 63, 66
PTF · 6

S

Setting the Activation Key · 8
Setting up Accounts · 20

T

Transmission Methods · 35, 36, 38, 39, 53, 55, 56, 62

U

Using in an ILE Program · 35, 65, 66, 77
Using in an OPM program · 38

W

WwwMrcSubmit · 23, 24, 35, 50, 53